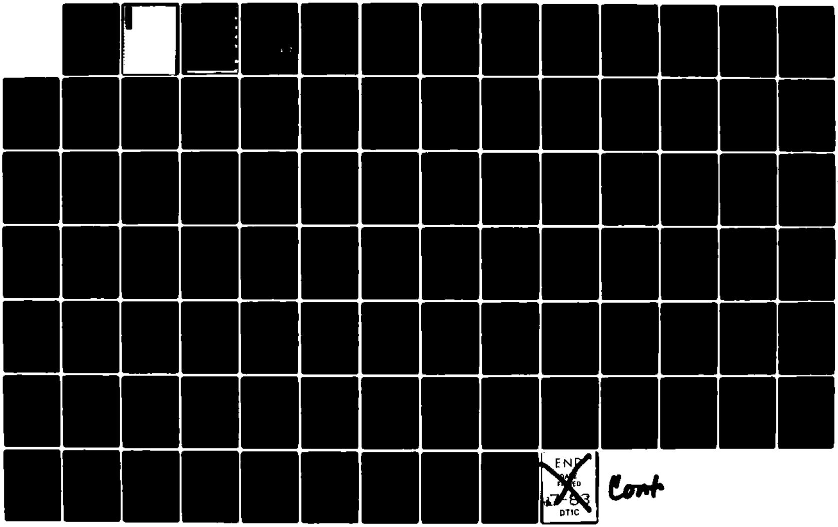
AD-A129 182   US ARMY INTELLIGENCE CENTER AND SCHOOL USAICS SOFTWARE
ANALYSIS AND MANAG..(U) JET PROPULSION LAB PASADENA CA
P BABBY ET AL. 09 JUL 82 JPL-D-181 NAS7-918

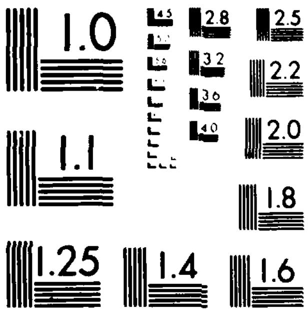
1/2

UNCLASSIFIED   F/G 8/2   NL



END
DATE
FILMED
7-83
DTIC

Cont



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

UNCLAS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | READ INSTRUCTIONS<br>BEFORE COMPLETING FORM |
|---|---|
| 1. REPORT NUMBER<br>UAA002 — 2. GOVT ACCESSION NO.<br>AD-A129 182 | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle)<br>U.S. ARMY INTELLIGENCE CENTER AND SCHCOL USAICS, Software Analysis and Management System Analysis of Geographic Transformation Algorithms. | 5. TYPE OF REPORT & PERIOD COVERED<br>FINAL<br>6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s)<br>JET PROPULSION LABORATORY, California Institute of Technology,<br>Pasadena, California | 8. CONTRACT OR GRANT NUMBER(s)<br>NAS7- 918 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS<br>Jet Propulsion Laboratory ATTN: 171-209<br>California Institute of Technology<br>4800 Oak Grove, Pasadena, CA 91109 | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS<br>RE 182<br>AMEND # 187 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS<br>Commander<br>USAICS ATTN: ATSI-CD-SF<br>Ft Huachuca, AZ 85613 | 12. REPORT DATE<br>JULY 9, 1982<br>13. NUMBER OF PAGES<br>35 |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)<br>(Same as ITEM 11)<br>Commander, USAICS.......... | 15. SECURITY CLASS. (of this report)<br>UNCLAS<br>15a. DECLASSIFICATION/DOWNGRADING SCHEDULE<br>NONE |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public dissemination

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Prepared by the Jet Propulsion Labs for the US Army Intelligence Center and School.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

GEOGRAPHIC TRANSFORMATI( ALGORITHM.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the findings of JPL regarding geographic transformation algorithms used in MAGIIC, Guardrail, Trailblazer and BETA systems. A set of parameters is developed to characterize and catalogue intelligence system algorithms in the 4 systems. Individual algorithms are also analyzed to determine if they are performing their functions properly.

UNCLAS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

D-181

# U.S. ARMY INTELLIGENCE CENTER AND SCHOOL

# USAICS

# Software Analysis and Management System

# Analysis of Geographic Transformation Algorithms

July 9, 1982

Approval:

Edward J Records for Fred Lesh

H. F. Lesh, Manager
USAICS Software Analysis and
Management System

John Radbill

John Radbill, Task Leader
Algorithm Analysis
USAICS Software Analysis and
Management System

Harry Jeane

H. Jeane, Manager
Software Development Section

A. Ferrari, Manager
Defense Information Systems Program

Concur:

William Akins

Major W. Akins, Chief
All Source Analysis System Office
U.S. Army Intelligence Center and School

JET PROPULSION LABORATORY
California Institute of Technology
Pasadena, California

UAA002

The following people contributed to this report

P. Babby

J. Gillis

A. Griesel

N. Palmer

J. Radbill

# TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

## 1. Introduction

### 1.1. Purpose

This report[*] describes the findings of the Algorithm Analysis Subtask group working on the U.S. Army Intelligence Center and School (USAICS) Software Analysis and Management System task (USAMS) regarding geographic transformation algorithms used in four of the intelligence-gathering systems under USAICS cognizance. In this report a set of parameters is developed to characterize and catalogue intelligence system algorithms in four specific systems. Individual algorithms are analyzed to determine whether they are performing their functions properly. Algorithms that perform the same function in different systems are compared to determine which ones are best according to various criteria.

The algorithms examined in this report are taken from the MAGIIC, Guardrail, Trailblazer, and BETA systems. They were chosen from the approximately 41 deployed intelligence systems for which USAICS is Combat Developer because their documentation was quickly accessible and because they represented a range of algorithm applications. Geographic transformation (mapping) algorithms were chosen for this report since all four systems contain position location/description functions and many of their algorithms are unclassified.

### 1.2. Background

Each of the about 41 intelligence systems under USAICS cognizance employs several types of algorithms to carry out its gathering and processing of intelligence data. Two important types of these algorithms, geographic transformation and correlation, have been chosen for analysis during this year. The former translates grid zone locations, for example, from latitude-longitude to Universal Transverse Mercator (UTM), while the latter resolves many individual sitings into militarily recognizable targets based chiefly on standard statistical procedures. It is important to develop a set of parameters to characterize these algorithms so that how they should be catalogued can be

[*] Two additional reports will be submitted in FY82: a correlation analysis report and a report on possible algorithm analysis methodologies.

determined. When these activities are completed, it becomes possible to compare algorithms that perform the same function in different systems.

To begin this process, the JPL Algorithm Analysis Sub-task group has examined the geographic transformation algorithms for four of the 41 systems, namely The Mobile Army Ground Imagery Interpretation Center (MAGIIC), Guardrail, Trailblazer, and Battlefield Exploitation and Target Acquisition (BETA). These four systems chosen for more detailed study represent several intelligence data analysis functions. MAGIIC is a ground-based analysis system to assist in interpreting hard-copy images from different airborne surveillance systems, including a capability for computerized mensuration on imagery; it can also receive and analyze data from Tactical Electronic Intelligence (TEREC) collection systems and provide emitter location estimates. Guardrail uses airborne sensor platforms to collect data on Direction Finding (DF) emitters; extensive ground-based software is then used to estimate the location of the units, such as command posts, associated with these emitters. Trailblazer also uses DF data to estimate emitter location. Its sensor platforms are essentially fixed and ground-based. BETA is a Test Bed program for correlating data received from several types of sensor systems and making target nominations. Both automatic correlation and aggregation techniques and interactive graphics are used in the operator's analysis. These systems would generally be employed at Division or Corps level or at an Air Force Tactical Air Control Element (TACE) or Allied Tactical Air Force (ATAF); target nominations and tactical situation reports would be available to commanders and their staffs from Brigade through Echelons Above Corps (EAC).

USAICS has cognizance of a large number of algorithms integral to intelligence-gathering systems in various stages of development and deployment. The state of "deployment" of algorithms in the USAICS inventory ranges from that of products of research contracts not yet implemented in any system to those in fielded systems such as Trailblazer or Guardrail. In the latter systems the algorithms are documented in design documents (narrative English and equations), and/or in machine readable design language, and in code. Often not all of these forms of documentation are available for any one system. For research algorithms not yet implemented actual code, or even detailed flow charts, may not be available; thus analysis must rely solely on mathematical descriptions.

"Algorithm" means any set of rules for carrying out a single conceptual operation on a set of data, such as transforming latitude-longitude coordinates to UTM or determining a position from a number of direction measurements taken at known points.[**] Algorithms are often hierarchical, lower-level algorithms often being used to describe higher-level algorithms and thereby illuminating their underlying logical structure. Thus, results from one algorithm may be data for another. USAICS is interested in algorithms performing intelligence data processing functions central to their systems' mission and those performing crucial support functions, such as geographic location, common to a number of systems. Data management or mathematical function algorithms, although vital to the efficient functioning of the systems, are not being treated in these first algorithm analyses.

## 1.3. User Benefits

These analyses can benefit users in several ways. First, a catalog of existing algorithms will help USAICS avoid having algorithms redeveloped for new systems from first principles. Second, analysis of individual algorithms may, in a few cases, identify deficiencies worth correcting on the next system revision. Third, and most important, the comparison of algorithms performing the same function in different systems can lead to identifying guidelines for developing and/or selecting algorithms to include in new and revised systems. Selected algorithms from the systems studied will begin to form a library of intelligence algorithms with associated computer subroutines that will be analogous to the Collected Algorithms of the Association for Computing Machinery (ACM). The creation of such a library is in the spirit of Ada[+], the Department of Defense language for embedded systems, and Ada's environment.

[**] These conceptual models should be describable, although their technical implementation is often significantly more complicated to present.

[+] Ada is a trademark of the Department of Defense

## 2. Analyzing the Algorithms

### 2.1. Early Steps

Since the Location and Movement Analysis System (LAMAS) system documentation was available first, our early analysis efforts were directed to that system. A preliminary analysis of a Shortest Path Algorithm was done anı modeled in Pascal as an approach to standardizing representations. This algorithm was a variant of Dijkstra's Shortest Path Algorithm.

Later the sponsor decided that our first emphasis should be on the coordinate conversion algorithms of the MAGIIC, Guardrail, Trailblazer and BETA systems. Three approaches to this analysis were tried and evaluated. The MAGIIC system has been hierarchically analyzed for the interrelationship of the algorithms. The spheroid models of the Earth's oblateness have been examined for all the systems. The grid zone generation algorithms have been compared across all four systems.

### 2.2. Learning Military Mapping

To analyze the first type of algorithm required learning the military grid system. This discussion identifies the various map projections and military grid reference systems examined and how they are interrelated. The scope of this discussion is limited to only those map projections and grid reference systems pertinent to the MAGIIC, BETA, Guardrail, and Trailblazer systems.

The map projections discussed are the Transverse Mercator, Polar Stereographic, Lambert Conformal Conic, and the Gnomonic. The grid reference systems used are the Universal Transverse Mercator (UTM), Universal Polar Stereographic (UPS), and Military Grid Reference System (MGR). The selection of a map projection is based on the properties it preserves in the transformation from a three-dimensional spheroid to a two-dimensional plane. These properties include orthogonality of latitude and longitude, equal area representation, distortion of shape, minimal change in scale factor in either east-west or north-south directions, and representation of great circles by straight lines. Since all map projections are from a spheroid model of the Earth, the parameters that the spheroid model use are very important. Various spheroid models are used for different portions of the Earth.

The selection of a grid reference system depends on the portion of the Earth examined and the resolution desired. The UTM and UPS coordinate systems were adopted as standards by the military to minimize coordination problems due to the proliferation of locally-used grid reference systems. These coordinate systems are most suitable for the representation of large geographic areas (greater than 9° in latitude and longitude). The MGR system provides greater resolution when representing smaller geographic areas (within 100,000-meter by 100,000-meter squares). The MGR system can be overlaid on the UTM and UPS coordinate systems to eliminate ambiguity due to repetitions of the 100,000-meter square identifiers. The geographic reference system is simply given as a longitude-latitude pair. However, this reference system of zones is cumbersome for representing locations in good resolution. Also, there is an inconsistency in the form of the coordinates: some applications use decimal degree notation while others use clock-like representations.

The UTM grid reference system is valid for all longitudes over latitudes between 84° North and 80° South. This area is divided into rectangles of 6° in longitude (zones) by 8° in latitude (bands), except for the 12° band from 72° to 84° latitude. There are 60 zones numbered from 1 through 60 for the zones from -180° to +180° longitude. There are 22 bands lettered C through X for the bands from -80° to 84° latitude. There are some subtle irregularities in this pattern beyond 56° latitude between 0° and 45° in longitude (see Figure 2-1). The UTM grid reference system is based on the Modified Transverse Mercator projection, but can be mathematically transformed for use with other types of projections.

The UPS grid reference system is valid for the North Polar (+84° longitude to the pole) and the South Polar regions (-80° longitude to the pole). These regions have a grid zone number of zero and consist only of a grid zone letter that is longitude-dependent. The North Polar region grid zone letters are Y (Western hemisphere) and Z (Eastern Hemisphere). The South Polar regions are A and B. The UPS grid reference system is based on the Polar Stereographic projection (see Figure 2-2).

The MGR grid reference system, illustrated for the UPS system in Figure 2-2 and for the UTM system in Figure 2-3, provides finer resolution than the UTM or UPS grid reference systems. It identifies 100,000-meter by 100,000-

meter squares by two letters, an Easting letter and a Northing letter. These letters are sequenced so as to provide at least 18° separation between similarily-lettered squares (within a given spheroid model area - otherwise the separation is 9°). These lettering sequences are biased and restarted at the boundaries of the underlying spheroid models.

These MGR system letter designations may be used without reference to the UTM or UPS designations when there is no likelihood of ambiguity, otherwise the UTM or UPS designation is included. Positions within these squares can be interpolated in tens of meters and are referred to as Easting and Northing terms representing distances rather than degrees.

The Transverse Mercator projection transforms the Earth's spheroid onto a cylinder secant to the Earth and perpendicular to its axis. This projection is used at latitudes within 84° North and 80° South. Scale linearity is correct at the two meridians cut by the cylinder (6° apart) and quite accurate in the band formed by them. Because of the vertical linearity this projection is particularily suitable to areas of interest in the North-South direction. This projection lends itself well to being overlaid with a rectangular grid reference system (such as the MGR system).

The Polar Stereographic projection transforms the Earth's spheroid onto a plane tangential to the Earth (at the pole, in our applications). This projection is used at latitudes beyond 84° North and from 80° South. Scale linearity decreases and equal area exaggeration increases as the distance from the pole increases. Latitude-longitude orthogonality is preserved at the meridian crossings. All circles of latitude are concentric, centered at the pole. Thus, this projection is useful for plotting radio waves and air navigation with a compass.

The Lambert Conformal Conic projection transforms the Earth's spheroid onto a cone parallel to the Earth's axis and secant to the earth at two latitudes referred to as the standard latitudes. The East-West scale linearity is correct at the two standard latitudes and is relatively accurate in the band between these latitudes. The projection preserves direction and shape quite well within and near the standard latitudes. Hence, the Lambert, Conformal Conic Projection is best suited to East-West measurements and is useful for air navigation. Also, all meridians are straight and intersect at the pole. This projection is most applicable to the mid-latitude region where the cone is secant to the Earth.

The Gnomonic projection transforms the Earth's spheroid onto a plane tangent to the Earth's at the point of interest. This projection is valid over all latitudes and longitudes. It has the quality of representing all great circle arcs on the projection as straight lines. Since electromagnetic waves travel the shortest distance route (great circle arc), the Gnomonic projection is ideally suited for the presentation of direction-finding lines of bearing.

## 2.3. Representing Algorithms in Standard Form

To compare algorithms across systems, all algorithms analyzed must be translated into a standard format. Algorithms in the systems analyzed had been coded in such diverse languages as assembly and structured FORTRAN so that translation into a common Higher-Order Language (HOL) became essential to searching for common and diverse features. Publication ALGOL was seen as an attractive candidate because it has been used in the collected algorithms of the ACM for algorithm description. However, audiences outside the Applied Mathematics, Numerical Analysis, and Computer Sciences communities are generally unfamiliar with ALGOL; and compilers for ALGOL 60 or ALGOL 68 are not readily available in this country.

Pascal, an ALGOL-like language, has been chosen for the primary representation language for the algorithms because it has many of the properties of ALGOL (structure, strong typing, etc.), it has become familiar to a wide audience, and high quality compilers are available on many computers including the Digital Equipment Corp. VAX and many microcomputers with CPM operating systems. The last point is important because the VAX will be used by both USAICS and JPL, and the microcomputers are similar to the word processors and personal computers at JPL and USAICS. Among the features of Pascal that contribute to its clarity are the command structures, such as "if-then-else" and "case", and the user-defined data types. However, separate compilations of procedures to support hierarchical descriptions of algorithms are an implementation-dependent extension rather than a basic feature of the language. Because of this and other problems Pascal provides at best an interim solution to the algorithm description problem.

Ada offers a long-term solution. The Ada language avoids many of the shortcomings of Pascal and has many additional features. A stronger reason for using Ada is that the Army is likely to require all new systems initiated after 1984 to be programmed in it. While no complete compiler for Ada is currently available, there is an interpreter on the project VAX computer, although it is very slow. A compiler for an incomplete implementation available on Z80-based microcomputers with CPM operating systems is also available. Although this compiler is not completely satisfactory because of the lack of user-defined types, it is still useful for some simple examples and for comparison with Pascal.

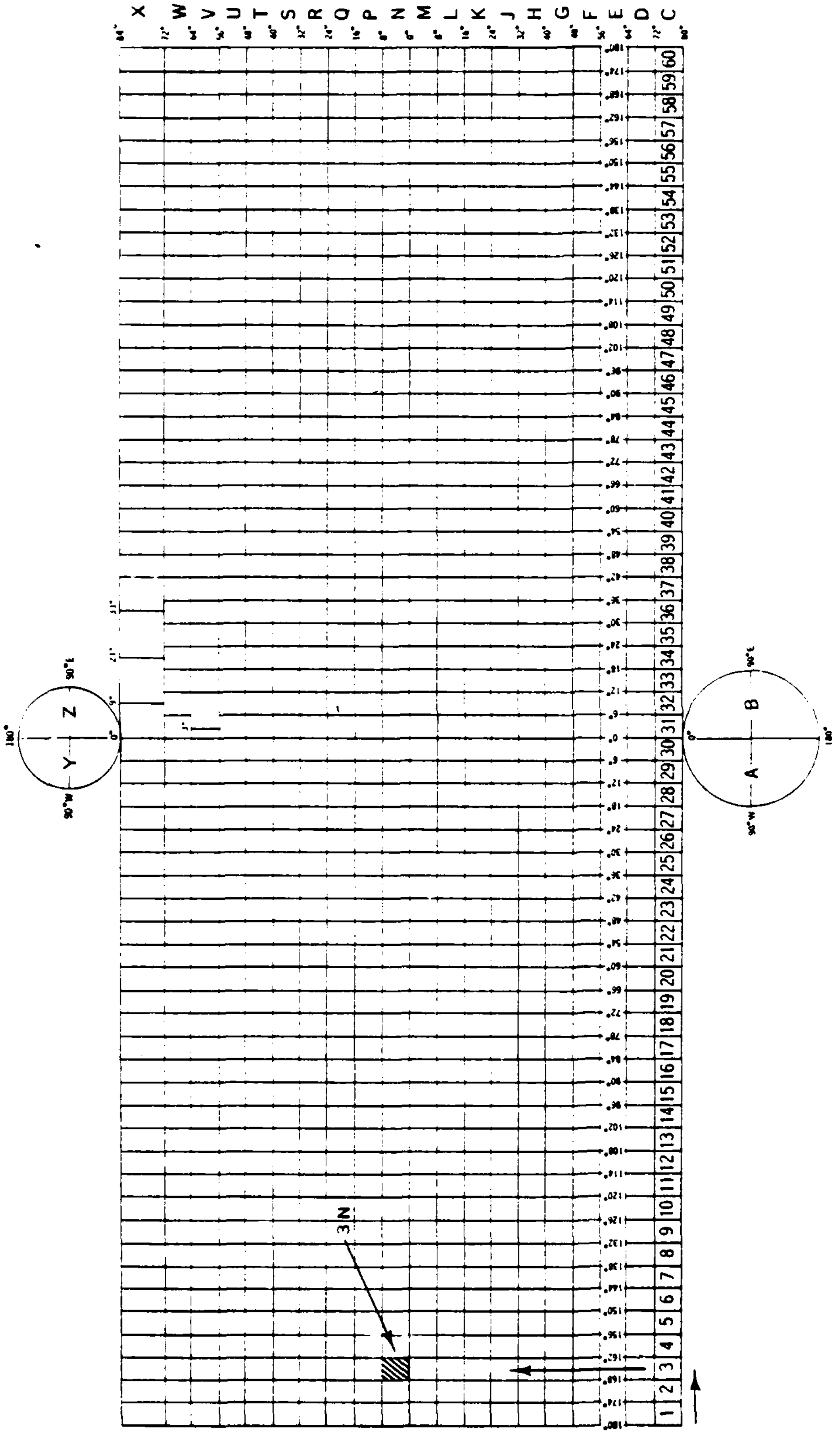
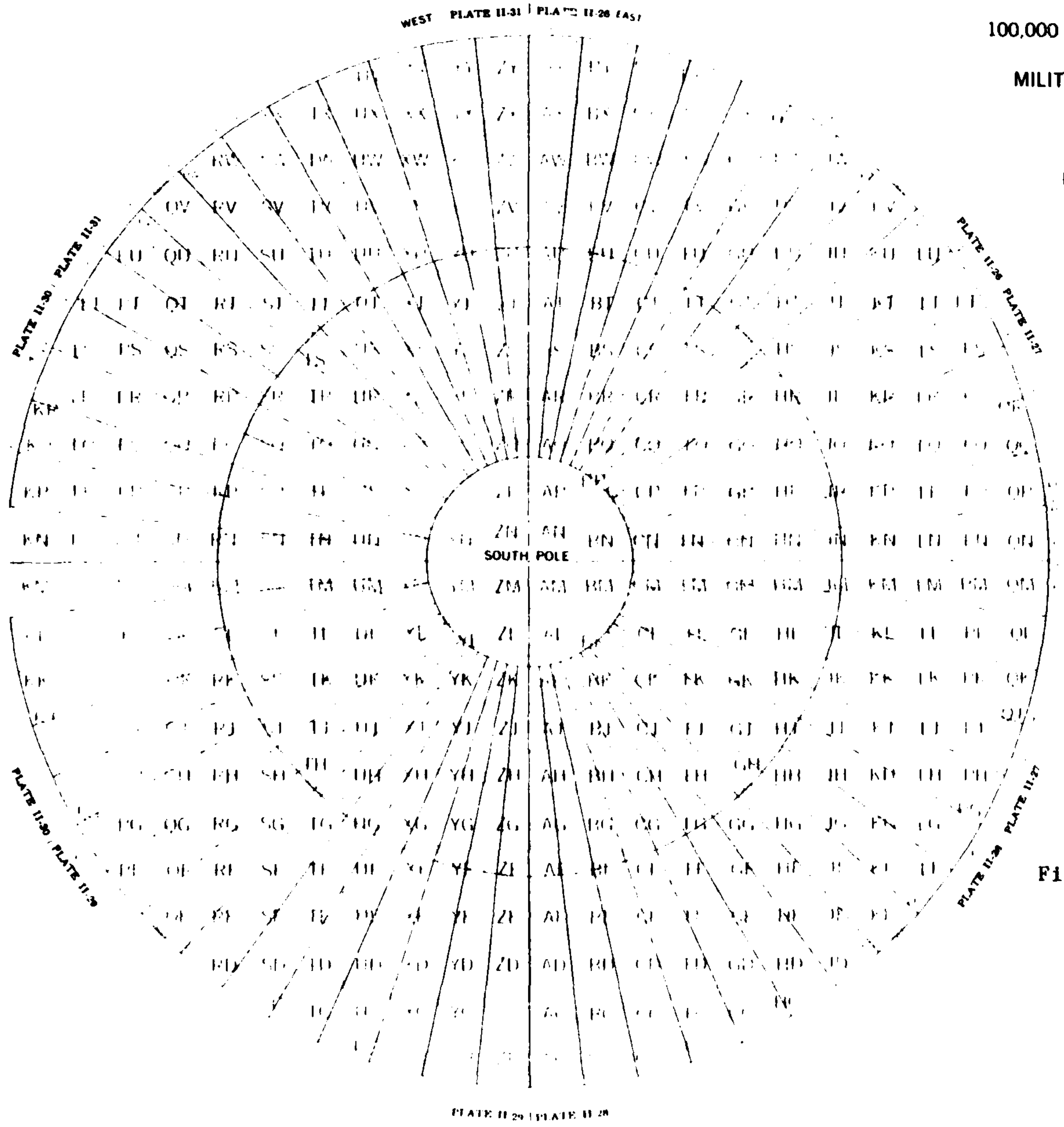


Figure 2-1: Grid Zone Designations of the Military Grid Reference System (UTM)

100,000 METER SQUARE IDENTIFICATIONS
FOR THE
MILITARY GRID REFERENCE SYSTEM

INTERNATIONAL ELLIPSOID


WEST PLATE 11-31 | PLATE 11-26 EAST
PLATE 11-30 | PLATE 11-31
PLATE 11-26 PLATE 11-27
SOUTH POLE
ZN
AN
ZM
AM
PLATE 11-30 | PLATE 11-29
PLATE 11-28 PLATE 11-27
PLATE 11-29 | PLATE 11-28

GRID ZONE DESIGNATIONS

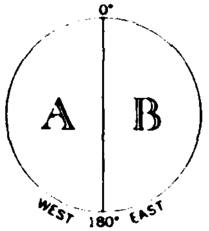
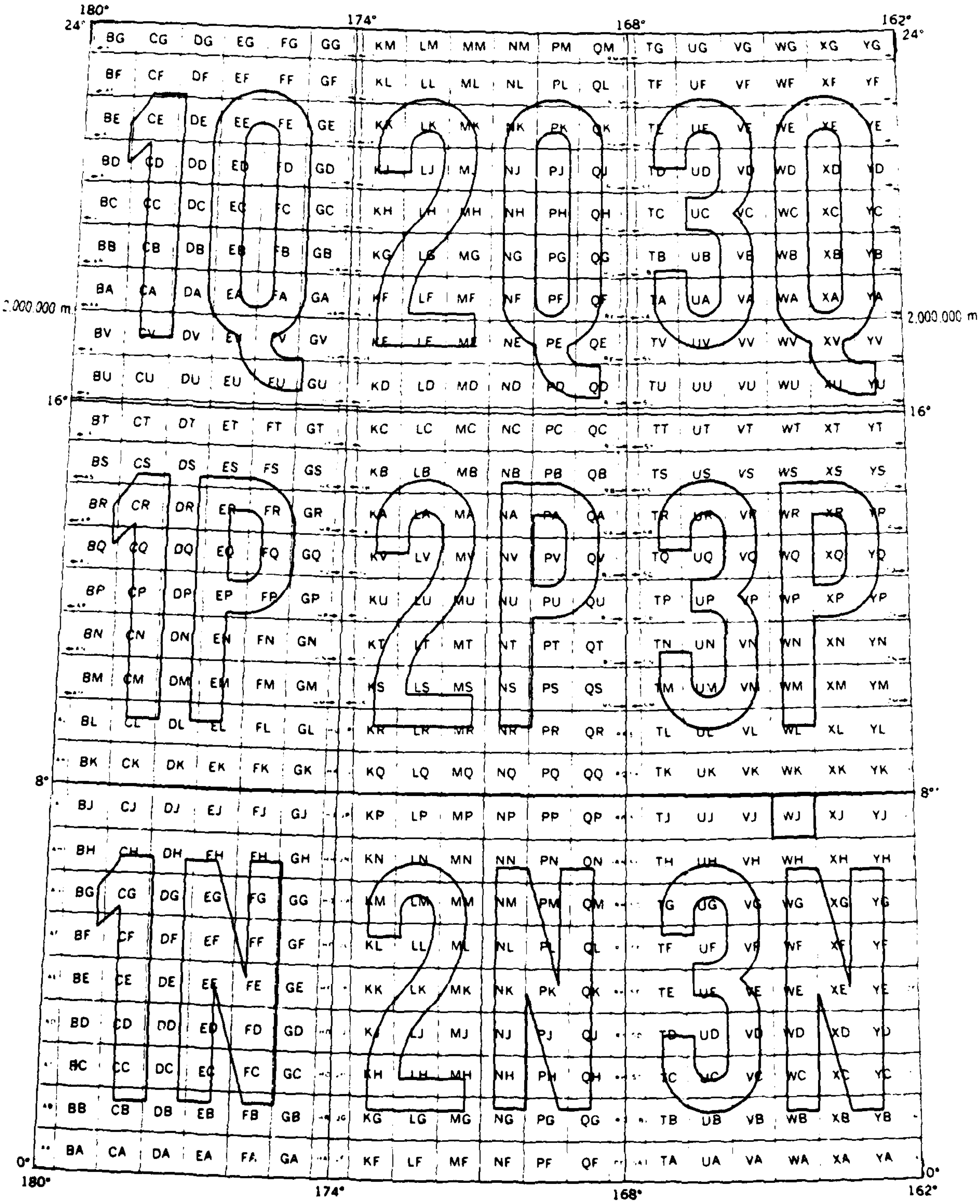


Figure 2-2: Grid Zone Designations of
Universal Polar Sterographic (UPS)



| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BG | CG | DG | EG | FG | GG | KM | LM | MM | NM | PM | QM | TG | UG | VG | WG | XG | YG |
| BF | CF | DF | EF | FF | GF | KL | LL | ML | NL | PL | QL | TF | UF | VF | WF | XF | YF |
| BE | CE | DE | EE | FE | GE | KK | LK | MK | NK | PK | QK | TE | UE | VE | WE | XE | YE |
| BD | CD | DD | ED | FD | GD | KJ | LJ | MJ | NJ | PJ | QJ | TD | UD | VD | WD | XD | YD |
| BC | CC | DC | EC | FC | GC | KH | LH | MH | NH | PH | QH | TC | UC | VC | WC | XC | YC |
| BB | CB | DB | EB | FB | GB | KG | LG | MG | NG | PG | QG | TB | UB | VB | WB | XB | YB |
| BA | CA | DA | EA | FA | GA | KF | LF | MF | NF | PF | QF | TA | UA | VA | WA | XA | YA |
| BV | CV | DV | EV | FV | GV | KE | LE | ME | NE | PE | QE | TV | UV | VV | WV | XV | YV |
| BU | CU | DU | EU | FU | GU | KD | LD | MD | ND | PD | QD | TU | UU | VU | WU | XU | YU |
| BT | CT | DT | ET | FT | GT | KC | LC | MC | NC | PC | QC | TT | UT | VT | WT | XT | YT |
| BS | CS | DS | ES | FS | GS | KB | LB | MB | NB | PB | QB | TS | US | VS | WS | XS | YS |
| BR | CR | DR | ER | FR | GR | KA | LA | MA | NA | PA | QA | TR | UR | VR | WR | XR | YR |
| BQ | CQ | DQ | EQ | FQ | GQ | KV | LV | MV | NV | PV | QV | TQ | UQ | VQ | WQ | XQ | YQ |
| BP | CP | DP | EP | FP | GP | KU | LU | MU | NU | PU | QU | TP | UP | VP | WP | XP | YP |
| BN | CN | DN | EN | FN | GN | KT | LT | MT | NT | PT | QT | TN | UN | VN | WN | XN | YN |
| BM | CM | DM | EM | FM | GM | KS | LS | MS | NS | PS | QS | TM | UM | VM | WM | XM | YM |
| BL | CL | DL | EL | FL | GL | KR | LR | MR | NR | PR | QR | TL | UL | VL | WL | XL | YL |
| BK | CK | DK | EK | FK | GK | KQ | LQ | MQ | NQ | PQ | QQ | TK | UK | VK | WK | XK | YK |
| BJ | CJ | DJ | EJ | FJ | GJ | KP | LP | MP | NP | PP | QP | TJ | UJ | VJ | WJ | XJ | YJ |
| BH | CH | DH | EH | FH | GH | KN | LN | MN | NN | PN | QN | TH | UH | VH | WH | XH | YH |
| BG | CG | DG | EG | FG | GG | KM | LM | MM | NM | PM | QM | TG | UG | VG | WG | XG | YG |
| BF | CF | DF | EF | FF | GF | KL | LL | ML | NL | PL | QL | TF | UF | VF | WF | XF | YF |
| BE | CE | DE | EE | FE | GE | KK | LK | MK | NK | PK | QK | TE | UE | VE | WE | XE | YE |
| BD | CD | DD | ED | FD | GD | KJ | LJ | MJ | NJ | PJ | QJ | TD | UD | VD | WD | XD | YD |
| BC | CC | DC | EC | FC | GC | KH | LH | MH | NH | PH | QH | TC | UC | VC | WC | XC | YC |
| BB | CB | DB | EB | FB | GB | KG | LG | MG | NG | PG | QG | TB | UB | VB | WB | XB | YB |
| BA | CA | DA | EA | FA | GA | KF | LF | MF | NF | PF | QF | TA | UA | VA | WA | XA | YA |

Figure 2-3: Basic Plan of the 100,000-meter Square Identifications of the U.S. Army Military Grid Reference System, between 84° N. and 80° S

## 3. Characterizing and Cataloging the Algorithms

The set of properties stored for algorithms in the database should characterize an algorithm for military application purposes without requiring that the algorithm itself be retrieved and examined. The algorithm property selection is influenced by the following ways the database will be used. The user may wish a general summary of what algorithms are in the database. A more likely use is to look for algorithms that perform a specified function, such as position location. In another dimension, the user may ask "What algorithms do we have in MAGIIC?". Cataloging properties should be independent for efficiency of description. Two of the properties chosen, performance and robustness, are not totally independent. Requirements performance is interpreted here as, "does the algorithm do what it says it does?". Lack of robustness is interpreted here as failure for special value or failure because of small errors in input data. Range checking of input variables is an important contributor to robustness. This is particularly important if the algorithm is to be used for more than one system where the calling programs can not be expected to protect it.

The Algorithm Level Help File; (Figure 3-1) taken from the Acquisition and Database Entry Prompting Tool (ADEPT) User's Guide, is examined and interpreted as a means for describing the present classification scheme. This information is provided for each system in which an algorithm is implemented. This set of parameters is likely to change as more experience is gained with its use.

Examples of PSA reports are shown in Appendix 7.1.

**Fig. 3-1: Algorithm Level Help File**

NAME is a name description of the algorithms function.

SYNONYM is the algorithms abbreviation (the same as its VAX file element name).

SOURCE:AUTHOR is the document from which the algorithm was taken and author, if known.

PROCESSING is what JPL has done with the algorithm, e.g. Pascal program tested.

MATH:FIELD is what mathematical field the algorithm is based on, e.g. least squares.

ROBUSTNESS is a measure of sensitivity to values of variables input to the algorithm, e.g. a transformation algorithm may produce an incorrect result with inputs of + 180° longitude.

TREE:LEVEL is a rough indication of location of the algorithm in the hierarchy of algorithms in a system.

REQUIREMENTS/PERFORMANCE Since the requirements documents for particular algorithms are generally not available, requirements must be derived from design documents or comments in code. This item describes how well the algorithm meets these requirements.

REFERENCE is a pointer to the VAX file containing the representation of the algorithm in standard form: Pascal and in some cases Ada. (These are given in Appendix 7.3).

## 4. Algorithms in MAGIIC

The MAGIIC System Lambert Constant Generation algorithm was analyzed and modeled in Pascal. This algorithm is required for analyzing and modeling the Geographic to Lambert/Polar Grid and Lambert/Polar Grid to Geographic conversions.

The MAGIIC system coordinate conversions were studied, and two interrelated algorithms were analyzed and modeled in Pascal. These were the Polar Grid to UPS and Northing and Easting to UTM conversions. Several inconsistencies have been noted (perhaps only because of the technical writing). Modeling these interrelated algorithms in Pascal led to the decision to use a nonstandard Pascal feature - the VAX Pascal external procedure capability. This was considered necessary to best maintain structural integrity, accuracy and clarity in the algorithm representation.

### 4.1. MAGIIC Lambert Constant Generation Algorithm

The MAGIIC Lambert Constant Generation Algorithm, described in document CG108100A, dated 23 October 1978, paragraph 3.2.119, page 210, has insufficient input parameters and a lack of detail on setting the hemisphere flags to implement the algorithm in Pascal without making several assumptions. If we assume the availability of the underlying spheroid parameters, the Lambert Constant Generation Algorithm performs satisfactorily. The hemisphere flag issue remains unsettled: are they north-south hemispheres, east-west hemispheres, or both? - all three selections make sense in different contexts.

This algorithm has been modeled in Pascal on the VAX computer for uniformity of presentation and for comparison and analysis. All assumptions are included in comments in this Pascal representation (see Appendix 7.3).

### 4.2. An Interrelated Set of MAGIIC Coordinate Conversion Algorithms

There are a set of four interrelated coordinate conversion algorithms for MAGIIC that warrant a consolidated discussion because although they are pair-wise reciprocal and criss-cross "call" each other, their input and output parameters are specified inconsistently. These algorithms viewed as reciprocal pairs are:

| | | | |
|---|---|---|---|
| 1. | Polar Grid to UPS | (paragraph 3.2.118) | reciprocal |
| 2. | UPS to Polar Grid | (paragraph 3.2.117) | reciprocal |
| 3. | Northing and Easting to UTM | (paragraph 3.2.86) | reciprocal |
| 4. | UTM to Northing and Easting | (paragraph 3.2.85) | reciprocal |

They can also be viewed as "criss-cross" calling algorithms as follows:

| | | | |
|---|---|---|---|
| 1. | Polar Grid to UPS | (paragraph 3.2.118) | each calls the other |
| 2. | Northing and Easting to UTM | (paragraph 3.2.86) | each calls the other |
| 3. | UPS to Polar Grid | (paragraph 3.2.117) | each calls the other |
| 4. | UTM to Northing and Easting | (paragraph 3.2.85) | each calls the other |

Analysis of these algorithms and their interrelationships has revealed the following inconsistencies:

1. the Polar Grid to UPS algorithm should have an output list consistent with the UPS to Polar Grid algorithms input list since they are reciprocal functions. The lists are not consistent;

2. the Northing and Easting to UTM algorithm and the UTM to Northing and Easting algorithm, similarily, have inconsistent output and input lists;

3. in both of the above cases the same problems are true when the algorithms are used in the "criss-cross call" sense.

These points will be discussed in greater detail below in the independent analysis of the four algorithms.

## 4.3. MAGIIC Polar Grid to Universal Polar Sterographic Algorithm

The MAGIIC Polar Grid to Universal Polar Sterographic (UPS) Algorithm, described in document CG108100A, dated 23 October 1978, paragraph 3.2.118, page 209, fails to produce the correct output data.

This algorithm first determines if the UPS or UTM grid reference system is applicable based on the input grid zone designation. The algorithm that handles the UTM conversion is discussed as part of the Northing and Easting to Universal Transverse Mercator conversion algorithm (paragraph 3.2.86 of the previously referenced document).

In an attempt to render this algorithm amenable to analysis and modeling in Pascal several assumptions have been made (based on our interpretation of the composite of various inferences from paragraph 3.2.86, .87, .117, .118). These assumptions are:

1. The input/output lists consist of decimal and integer numbers, and alphabetic letters. There are no wholesale conversions of the input/output lists from and to ASCII representation,

2. The input/output lists are exchanged with the Northing and Easting to UTM algorithm for processing, when the input grid zone letter is from C thrugh X (not in either polar region). Otherwise, this algorithm performs the processing (for the polar regions).

The following discussion only treats the UPS coverage area where a conversion from PS to UPS should be made. This conversion is handled within the Polar Grid to Universal Polar Sterographic conversion algorithm (paragraph 3.2.118). This algorithm is defective in producing disallowed 100,000 meter squares letter pairs.

The North Polar Area ($\geq$ 84° N) and South Polar Area (> 80° S) are referenced using the UPS grid zone reference system with the grid zone number set to zero followed by one of the grid zone letters A, B, Y, or Z. Letter pairs represent the 100,000-meter squares which overlay the grid zone designations. It is in the lettering of these squares that this algorithm fails.

The descriptions of how to obtain the Easting letter $L_E$ from the index $I_E$ and the Northing letter $L_N$ fron the index $I_N$ are merely statements that the functions should be performed without any details. Perhaps an elaborated description of the details of these functions would lead to satisfactory conversions, if these details were furnished.

### 4.4. MAGIIC Northing and Easting to Universal Transverse Mercator Algorithm

The MAGIIC Northing and Easting to Universal Transverse Mercator (UTM) Alorithm as described in document CG18100A, dated 23 October 1978, paragraph 3.2.86, page 156, appears to produce the correct output data on the UTM map segment available for reference. It will certainly fail in grid zones 31V, 32X, 34X, and 36X and should be further evaluated.

This algorithm first determines, whether the UTM or UPS grid reference system is applicable based on the iput grid zone designation. The algorithm that handles the UPS conversion is discussed as part of the Polar Grid to Universal Polar Sterographic conversion algorithm (paragraph 3.2.118 of the previously referenced document).

In an attempt to render the algorithm amenable to analysis and to model in Pascal several assumptions have been made (based on our interpretation of the composite of various inferences from paragraph 3.2.86, .87, .117, .118). These assumptions are:

1. The input lists consist of decimal and integer numbers and alphabetic letters. There is no wholesale conversion of the in·ut list to ASCII representation. The output lists are entirely in ASCII representation.

2. The input/output lists are exchanged with the Polar Grid to UPS algorithm for processing, when the input grid zone letter is not from C through X (for the polar regions). Otherwise, this algorithm performs the processing (for the non-polar regions).

Only the conversion to the UTM coverage area, that is, the actual conversion handled within this Northing and Easting to Universal Transverse Mercator conversion algorithm (paragraph 3.2.86), is discussed here. This algorithm is defective in producing grid zone designation 31V, which should be truncated at 3° E, and the three non-existent grid zone designations 32X, 34X, and 36X.

## 5. Comparing the Algorithms Across Systems

### 5.1. Spheroid Models

Because the Earth is not a perfect sphere, it is modeled as a spheroid. Since this underlying spheroid model of the Earth's oblatness spans most of the coordinate conversion algorithms, the use of various spheroid models was investigated for all four systems. It was found that twelve different spheroid models were used among or in the four systems raising the possibility of inconsistencies that may hamper inter-system communication. Of these spheroid models five were used in common by all four systems. The remaining seven spheroid models were not available in all systems, as illustrated in Table 1. Some of these partially-shared spheroid models may be equivalent, but this cannot be determined until the code is available for all four systems.

### 5.2. Grid Zone Generation

The Grid Zone Generation algorithms were analyzed for the MAGIIC, Guardrail, Trailblazer, and BETA systems. The Guardrail Grid Zone algorithm text description is consistent with the Trailblazer text description and computer code. The MAGIIC text description differs from the Guardrail and Trailblazer descriptions and would tend to produce less efficient runtime code, but would economize memory. All three of these systems' algorithms would produce the same flawed results, except BETA which fails badly at the upper latitudes. These three versions of the Grid Zone Generation algorithm have been modeled in Pascal.

The BETA Grid Zone Generation algorithm handles details of grid zone generation more completely. The grid zone number calculations handle input longitude beyond -180° and at 180° and beyond whereas the three other systems would fail. The grid zone letter calculations handle the special conditions of grid zone truncation for grid description 31V and the non-existence of grid designations 32X, 34X, and 36X.

Pascal implementations of these algorithms are given in Appendix 7.3. The underlying assumptions are given in the comments included in the code.

## 5.3. MAGIIC Grid Zone Generation Algorithm

The MAGIIC Grid Zone Generation Algorithm as described in document CG108100A, dated 23 October 1978, paragraph 3.2.90, page 167 has been analyzed and found to handle the following five areas incorrectly:

1. the upper limit of longitude (180° E),

2. the latitudes $\geq$ 80° N (considerably below the upper limit, 84° N), where it provides erroneous data,

3. the truncated grid zone 31V,

4. the non-existent grid zones 32X, 34X, and 36X,

5. the regions beyond the stated longitude and latitude limits, where it fails catastrophically.

In general, perhaps due to technical writing, there are many errors of omission and/or commission where the criteria for certain algorithm parts are left unstated; for example, a text states, "If the latitude is 84° north or greater, or 80° south or greater, the grid zone number (sic) shall be set to Y or Z or to A or B, respectively. The grid zone number shall be set to zero."

## 5.4. Guardrail Grid Zone Generation Algorithm

The Guardrail Grid Zone Generation Algorithm as described in document ESL-TM928, dated 15 Septemer 1979, paragraph 16.6.2.1, page 16-192 fails in five areas:

1. at longitudes equal to and beyond 180° E and beyond 180° W.

2. at latitudes equal to and beyond 80° N and beyond 80° S,

3. at the truncated grid zone 3IV,

4. at the non-existent grid zones 32X, 34X, and 36X,

5. beyond the stated longitude and latitude limits, where it fails catastrophicaly.

## 5.5. Trailblazer Grid Zone Generation Algorithm

The Trailblazer Grid Zone Generation Algorithm is described in the well-commented ROLM assembly language listings. This algorithm, extracted from the code for the GP2UM subprogram dated 20 February 1981, fails in five areas:

1. at longitudes equal to and beyond 180° E and beyond 180° N,

2. at latitudes equal to and beyond 8°0 N and beyond 80° A,

3. at the truncated grid zone 31V,

4. at the non-existent grid zones 32X, 34X, and 36X,

5. beyond the stated longitude and latitude limits.

Because the hierarchical program structure is not yet available for Trailblazer, it is possible that some or all of these problems are handled adequately in higher levels of the program structure.

## 5.6. BETA Grid Zone Generation Algorithm

The BETA Grid Zone Generation Algorithm, described in document SS22-43, Appendix IV, page II-474 for the ADSONU subprogram, and page II-45D for the ADSCCM subprogram, was in Structured FORTRAN with in-line coding. The "INCLUDE" subprogram ZDBPRO was missing so some "reasonable" assumptions were made about it.

This algorithm performs as specified and effectively handles the following:

1. Grid zone wrap-around (longitudes >180° W or $\geq$ 180° E),

2. North and South Polar Regions (latitudes $\geq$ 84° N or > 80° S),

3. Truncating grid zone 31V,

4. The non-existent grid zones 32X, 34X, and 36X.

**Table 5-1: Inconsistent Spheroid Usage**

(X = spheroid model used in the system)

| Spheroid Model | System | | | | Technical Manual |
|---|---|---|---|---|---|
| | BETA | MAGIIC | GR | TB | TM241-1 |
| Clark 1866 | X | X | X | X | X |
| International | X | X | X | X | X |
| Clark 1880 | X | X | X | X | X |
| Everest | X | X | X | X | X |
| Bessel | X | X | X | X | X |
| Australian | X | X | X | | X |
| Walbeck | | | | X | |
| Fisher | X | | | X | |
| Krasovsky | | | X | X | |
| World Geodetic | X | | X | | |
| Airy | X | | | | |
| Malayan | X | | | | |
| Reference | 1 | 2 | 3 | 4 | |

1. DD2642, dtd 20 Feb 81, pg 263
2. CG1808100A, dtd 23 Oct 78, pg 168
3. ESL-TM929, dtd 15 Sep 79, pg 15-158
4. TM32-5811-022-10-0

## 6. Discussion and Conclusions

### 6.1. Documentation of Algorithms

Only a design document has been available for the MAGIIC system; and the code has not been available, as shown in Figure 6-1. Therefore, some of the apparent deficiencies in the algorithms may be due to poor technical writing and may not exist in the code itself. For the Guardrail system, tapes containing code have been available, but the printouts obtained to date have been largely unreadable so that suppositions made from the documentation could not be confirmed from the code. In the case of Trailblazer, code is available, but the structured overview expected from documentation is not available.

### 6.2. Similarity of Functions Across Systems

The functions performed by the geographic transformation algorithms are found to be basically the same across the four systems examined, although the functions are implemented in slightly different ways.

### 6.3. Incompleteness of Algorithms for Global Applications

The MAGIIC, Guardrail, and Trailblazer transformation algorithms do not account for all the vagaries of the military grid system. Only the BETA algorithms account for all regions and boundaries. The former systems may ensure that "bad" arguments are never passed to these algorithms, so that no anomalies would occur in overall system performance. However, to develop a library of algorithms shared by many systems requires that algorithms internally protect themselves from "bad" input data.

### 6.4. Robustness of Algorithms

All systems but BETA fail to check for limits of latitude and longitude. This may arise from a common tendency to focus attention on certain areas of the world, e.g. Western Europe. This tendency is especially inappropriate when developing software that may well outlive any given political or geographical constraints.

### 6.5. Selection/Consolidation of Algorithms

The BETA grid zone generation algorithm is superior to those in the other three systems. Selection of a spheroid model for the library is not possible on the basis of the presently available data and may eventually require developing algorithms based on our experience with many different systems.

**Fig. 6-1: Documentation of Algorithms**

| System | Documentation | Code Available | Comments |
|---|---|---|---|
| MAGIIC | Yes<br>(Barely usable) | No | Bad technical writing: Errors of omission and bad-quality copy. Portions of some pages unreadable. TEREC task similar to Guardrail. |
| Guardrail | Yes<br>Good | No<br>Glimpses of code appear to be structured FORTRAN; most sections are missing | Multiple Tasks:<br>Program structure "implied" by document's structure. Flowcharts with verbal description. At least one significant technique covered by math description only, entirely included in one box at the flowchart level. |
| Trailblazer | No<br>Not separately published, but basic documentation included at the beginning of each code segment | Yes<br>assembly | Well commented code.<br>Many similarities to Guardrail. |
| BETA | Yes<br>Good | Yes<br>Comprehensive Structured FORTRAN | Program structure explicitely included in documents. Some "key" charts not readable due to photo reductions. Code is in-line commented from Program Design Language (PDL), but we don't have the PDL |

## 7. Appendices

### 7.1. Database Entries and Products (PSL/PSA)

The three attached PSA Reports were found to be useful to our analysis. The first report is essentially the PSA Report representing our PSL input data descriptions. The second is the PSA Data Activity Interaction Matrix. It shows the interrelationships between the algorithms and their input (R) and output (D) data items. The third is part of the PSA Structure Chart and is in indented hierarchy chart for the algorithms in our PSL database.

## 7.2. Algorithm Hierarchy Charts

Structure Report

```
                                                                 28 MG_STEROHGHT --------------- /
                                                                27 MG_PROXSRCH --------------- / |
                                                             26 MG_NE2MAPCUR --------------- / | |
                                                                                             | | |
                                                          25 MG_PRIPTDET --------------- /   | | |
                                                      24 MG_RMSTARGLOC --------------- / |   | | |
                                                     23 MG_PRECPTFLE --------------- / | |   | | |
                                                     22 MG_NAVCORN --------------- / | | |   | | |
                                                 21 MG_LAMBERTCG --------------- / | | | |   | | |
                                                                                 | | | | |   | | |
                                               20 MG_MAPITZN --------------- /   | | | | |   | | |
                                            19 MG_LORANCNV --------------- / |   | | | | |   | | |
                                        18 MG_IICOORDCNV --------------- / | |   | | | | |   | | |
                                        17 MG_RMSINVTL --------------- / | | |   | | | | |   | | |
                                       16 MG_IIINVTL --------------- / | | | |   | | | | |   | | |
                                                                     | | | | |   | | | | |   | | |
                                    15 MG_PG2UPS --------------- /   | | | | |   | | | | |   | | |
                                  14 MG_NE2UTM --------------- / |   | | | | |   | | | | |   | | |
                                13 MG_UTM2NE --------------- / | |   | | | | |   | | | | |   | | |
                            12 MG_FRAMSRCH --------------- / | | |   | | | | |   | | | | |   | | |
                             11 MG_NE2GP --------------- / | | | |   | | | | |   | | | | |   | | |
                                                         | | | | |   | | | | |   | | | | |   | | |
                    10 MG_GRDZONECNV --------------- /   | | | | |   | | | | |   | | | | |   | | |
                      9 MG_GP2L/PG --------------- / |   | | | | |   | | | | |   | | | | |   | | |
                      8 MG_GP2NE --------------- / | |   | | | | |   | | | | |   | | | | |   | | |
                7 MG_MAPCUR2NE --------------- / | | |   | | | | |   | | | | |   | | | | |   | | |
              6 MG_IIFLMITZN --------------- / | | | |   | | | | |   | | | | |   | | | | |   | | |
                                             | | | | |   | | | | |   | | | | |   | | | | |   | | |
         5 MG_BUILDSKTCH --------------- /   | | | | |   | | | | |   | | | | |   | | | | |   | | |
        4 MG_GENSPHERE --------------- / |   | | | | |   | | | | |   | | | | |   | | | | |   | | |
      3 MG_IITARGLOC --------------- / | |   | | | | |   | | | | |   | | | | |   | | | | |   | | |
      2 MG_BUILDCD --------------- / | | |   | | | | |   | | | | |   | | | | |   | | | | |   | | |
 1 MG_Algorithms --------------- / | | | |   | | | | |   | | | | |   | | | | |   | | | | |   | | |
                                 | | | | |   | | | | |   | | | | |   | | | | |   | | | | |   | | |
-------------------------------+-----------+-----------+-----------+-----------+-----------+-------+
MG_Algorithms ---------------- |   A     A |           |   A       |     A A A |   A A   A |   A A |
MG_BUILDCD ------------------- |     B     |           |           |           |           |       |
MG_IITARGLOC ----------------- |       B   |           |           |           |           |       |
MG_GENSPHERE ----------------- |           |           |           |           |           |       |
MG_BUILDSKTCH ---------------- |     B     | B B       |           |           |           |       |
                               +-----------+-----------+-----------+-----------+-----------+-------+
MG_IIFLMITZN ----------------- |           |           |           |           |           |       |
MG_MAPCUR2NE ----------------- |           |     B   B | B         |           |           |       |
MG_GP2NE --------------------- |       B   |       B B |           |           |           |       |
MG_GP2L/PG ------------------- |           |           |           |           |           |       |
MG_GRDZONECNV ---------------- |           |           |           |           |           |       |
                               +-----------+-----------+-----------+-----------+-----------+-------+
MG_NE2GP --------------------- |           |     B     |           |           |           |       |
MG_FRAMSRCH ------------------ |           |           | B   B     | B B       |           |       |
MG_UTM2NE -------------------- |       B   |           | B     B   |           |           |       |
MG_NE2UTM -------------------- |           |           |         B |           |           |       |
MG_PG2UPS -------------------- |           |           |       B   |           |           |       |
                               +-----------+-----------+-----------+-----------+-----------+-------+
MG_IIINVTL ------------------- |     B B   |           |           |           |           |       |
MG_RMSINVTL ------------------ |           |           |           |           |           |       |
MG_IICOORDCNV ---------------- |           |     B     | B   B B   |           |           |       |
MG_LORANCNV ------------------ |       B   |           |           |           |           |       |
MG_MAPITZN ------------------- |           |           |           |           | B         |       |
                               +-----------+-----------+-----------+-----------+-----------+-------+
MG_LAMBERTCG ----------------- |           |           |           |           |           |       |
MG_NAVCORN ------------------- |     B     | B B B     | B   B B   |           |           |       |
MG_PRECPTFLE ----------------- |           |           |           |           |       B   |       |
MG_RMSTARGLOC ---------------- |           |           |           |           |           |       |
MG_PRIPTDET ------------------ |     B     | B B B     | B   B B   | B B       |       B   | B     |
                               +-----------+-----------+-----------+-----------+-----------+-------+
MG_NE2MAPCUR ----------------- |           |     B B   | B         |           |           |       |
MG_PROXSRCH ------------------ |           |   B B     |     B     |           |           |       |
MG_STEROHGHT ----------------- |     B     | B         |           | B         |           |       |
-------------------------------+-----------+-----------+-----------+-----------+-----------+-------+
```

Version A5.2R0

SEMANTIC-MAP-TEST

Utilizes Matrix

```
                                                                27 MG_NE2MAPCUR ------------------ /
                                                             26 MG_RMSTARGLOC ------------------ / |
                                                                                                 | |
                                                           25 MG_LAMBERTCG ------------------ /  | |
                                                            24 MG_PG2UPS ------------------ / |  | |
                                                          23 MG_NE2UTM ------------------ / | |  | |
                                                      22 MG_RMSINVTL ------------------ / | | |  | |
                                                     21 MG_IIINVTL ------------------ / | | | |  | |
                                                                                      | | | | |  | |
                                                   20 MG_UTM2NE ------------------ /  | | | | |  | |
                                                19 MG_GP2L/PG ------------------ / |  | | | | |  | |
                                           18 MG_GRDZONECNV ------------------ / | |  | | | | |  | |
                                              17 MG_NE2GP ------------------ / | | |  | | | | |  | |
                                            16 MG_GP2NE ------------------ / | | | |  | | | | |  | |
                                                                           | | | | |  | | | | |  | |
                                     15 MG_MAPCUR2NE ------------------ /  | | | | |  | | | | |  | |
                                   14 MG_IIFLMITZN ------------------ / |  | | | | |  | | | | |  | |
                                 13 MG_GENSPHERE ------------------ / | |  | | | | |  | | | | |  | |
                               12 MG_IITARGLOC ------------------ / | | |  | | | | |  | | | | |  | |
                             11 MG_STEROHGHT ------------------ / | | | |  | | | | |  | | | | |  | |
                                                                | | | | |  | | | | |  | | | | |  | |
                           10 MG_PROXSRCH ------------------ /  | | | | |  | | | | |  | | | | |  | |
                          9 MG_PRIPTDET ------------------ / |  | | | | |  | | | | |  | | | | |  | |
                       8 MG_PRECPTFLE ------------------ / | |  | | | | |  | | | | |  | | | | |  | |
                       7 MG_NAVCORN ------------------ / | | |  | | | | |  | | | | |  | | | | |  | |
                     6 MG_MAPITZN ------------------ / | | | |  | | | | |  | | | | |  | | | | |  | |
                                                     | | | | |  | | | | |  | | | | |  | | | | |  | |
                 5 MG_LORANCNV ------------------ /  | | | | |  | | | | |  | | | | |  | | | | |  | |
             4 MG_IICOORDCNV ------------------ / |  | | | | |  | | | | |  | | | | |  | | | | |  | |
             3 MG_FRAMSRCH ------------------ / | |  | | | | |  | | | | |  | | | | |  | | | | |  | |
         2 MG_BUILDSKTCH ------------------ / | | |  | | | | |  | | | | |  | | | | |  | | | | |  | |
          1 MG_BUILDCD ------------------ / | | | |  | | | | |  | | | | |  | | | | |  | | | | |  | |
                                          | | | | |  | | | | |  | | | | |  | | | | |  | | | | |  | |
----------------------------------------+----------+----------+----------+----------+----------+----+
 1 MG_Algorithms ---------------------- | S S S S S| S S S S S| S        |          |          |    |
 2 MG_BUILDCD ------------------------- |          |          |   U      |          |          |    |
 3 MG_IITARGLOC ----------------------- |          |          |     U    |          |          |    |
 4 MG_BUILDSKTCH ---------------------- |          |          |   U   U U|          |          |    |
 5 MG_MAPCUR2NE ----------------------- |          |          |          | U U U    |          |    |
----------------------------------------+----------+----------+----------+----------+----------+----+
 6 MG_GP2NE --------------------------- |          |          |     U    |     U U  |          |    |
 7 MG_NE2GP --------------------------- |          |          |          | U        |          |    |
 8 MG_FRAMSRCH ------------------------ |          |          |          |   U     U| U U      |    |
 9 MG_UTM2NE -------------------------- |          |          |     U    |   U      |     U    |    |
10 MG_NE2UTM -------------------------- |          |          |          |          |       U  |    |
----------------------------------------+----------+----------+----------+----------+----------+----+
11 MG_PG2UPS -------------------------- |          |          |          |          |     U    |    |
12 MG_IIINVTL ------------------------- |          |          |   U U    |          |          |    |
13 MG_IICOORDCNV ---------------------- |          |          |          | U U     U|     U    |    |
14 MG_LORANCNV ------------------------ |          |          |     U    |          |          |    |
15 MG_MAPITZN ------------------------- |          |          |          |          |         U|    |
----------------------------------------+----------+----------+----------+----------+----------+----+
16 MG_NAVCORN ------------------------- |          |          |   U   U U| U U     U|     U    |    |
17 MG_PRECPTFLE ----------------------- |          |          |          |          |          | U  |
18 MG_PRIPTDET ------------------------ |          |          |   U   U U| U U     U| U U U    | U U|
19 MG_NE2MAPCUR ----------------------- |          |          |          | U U   U  |          |    |
20 MG_PROXSRCH ------------------------ |          |          |         U| U       U|          |    |
----------------------------------------+----------+----------+----------+----------+----------+----+
21 MG_STEROHGHT ----------------------- |          |          |   U   U  |          | U        |    |
----------------------------------------+----------+----------+----------+----------+----------+----+
```

Data Activity Interaction Matrix

```
                   34 MG_UTM2NE -------------------- /
                 33 MG_UFS2FG ------------ ------- / |
               32 MG_STEROHGHT ----------------- / | |
             31 MG_RMSTARGLOC ---------------- / | | |
                                               | | | |
          30 MG_RMSINVTL ------------------ /  | | | |
        29 MG_RMSFLMITZN ---------------- / |  | | | |
      28 MG_RMSCOORDCNV --------------- / | |  | | | |
    27 MG_RAD2DEG ------------------- / | | |  | | | |
  26 MG_FROXSRCH ------------------ / | | | |  | | | |
                                    | | | | |  | | | |
----------------------------------+----------+--------+
 1 MG_Angle_in_Degrees/Mins/Secs  |   D      |        |
 2 MG_Angle_in_Scaled_Pi_Radians  |   R      |        |
 3 MG_F75T2 --------------------- |          |        |
 4 MG_Spheroid ------------------ |          |     D R|
 5 MG_F171TS -------------------- |          |        |
                                  +----------+--------+
 6 MG_Latitude/Longitude -------- |          |        |
 7 MG_Northing/Easting ---------- |          |     D F|
 8 MG_Grid_Zone_No./Letter ------ |          |     D R|
 9 MG_xy_Map_Cursor_Coordinate -- |          |        |
10 MG_UTM_Northing/Easting_Set -- |          |        |
                                  +----------+--------+
11 MG_Grid_Zone_Letter ---------- |          |     R  |
12 MG_N/E_Letters --------------- |          |     R  |
13 MG_N/E_Numbers --------------- |          |     R  |
14 MG_100K_Meter_Square_ID ------ |          |       R|
----------------------------------+----------+--------+
```

Data Activity Interaction Matrix

25 MG_PRIPIDET
24 MG_PRECPTFLE
23 MG_PG2UPS
22 MG NE2UTM
21 MG_NE2MAPCUR

20 MG_NE2GP
19 MG_NAVCORN
18 MG_MAPIT7N
17 MG_MAPCUR2NE
16 MG_LORANCNV

15 MG_LAMBERTCG
14 MG_L/PG2GP
13 MG_IITARGLOC
12 MG_IIINVTL
11 MG_IIFLMITZN

10 MG_IICOORDCNV
9 MG_GRDZONECNV
8 MG_GP2NE
7 MG_GP2L/PG
6 MG_GENSPHERE

5 MG_FRAMSRCH
4 MG_DEG2RAD
3 MG_BUILDSKTCH
2 MG_BUILDCD
1 MG BEACONF

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 MG_Angle_in_Degrees/Mins/Secs | | | | R | | | | | | | | | | | | | | | | | | | | | |
| 2 MG_Angle_in_Scaled Pi_Radians | | | | D | | | | | | | | | | | | | | | | | | | | | |
| 3 MG_F75T2 | | | | | | R | R | R | | | | | | R | R | | R | | | R | R | | | | |
| 4 MG_Spheroid | | | | | | D | | R | | | | | | R | | | D | | | R | R | R | R | | |
| 5 MG_F171T5 | | | | | | R | | | | | | | | | | | | | | | | | | | |
| 6 MG_Latitude/Longitude | | | | | | R | R | R | R | | | | | D | | | | | | D | R | | | | |
| 7 MG Northing/Easting | | | | | | | D | D | | | | | | R | | | D | | | R | R | R | R | | |
| 8 MG Grid Zone_No./Letter | | | | | | | D | R | D | | | | | R | | | R | | | R | R | | R | | |
| 9 MG_[illegible]_Map_Cursor_Coordinate | | | | | | | | | | | | | | | | | R | | | | D | | | | |
| 10 MG_UTM_Northing/Easting Set | | | | | | | | | | | | | | | | | | | | | | D | | | |
| 11 MG_Grid_Zone_Letter | | | | | | | | | | | | | | | | | | | | | | | D | | |
| 12 MG N/E Letters | | | | | | | | | | | | | | | | | | | | | | | D | | |
| 13 MG_N/E Numbers | | | | | | | | | | | | | | | | | | | | | | | D | | |
| 14 MG_100k Meter_Square ID | | | | | | | | | | | | | | | | | | | | | | | | | |

```
   1 MG_Algorithms
    2   MG_BUILDCD
     3     MG_IITARGLOC
      4       MG_GENSPHERE
    2   MG_BUILDSKTCH
     3     MG_IITARGLOC
      4       MG_GENSPHERE
     3     MG_IIFLMITZN
     3     MG_MAPCUR2NE
      4       MG_GP2NE
       5         MG_GENSPHERE
       5         MG_GP2L/PG
       5         MG_GRDZONECNV
      4       MG_GRDZONECNV
      4       MG_NE2GP
       5         MG_GP2NE
        6           MG_GENSPHERE
        6           MG_GP2L/PG
        6           MG_GRDZONECNV
    2   MG_FRAMSRCH
     3     MG_NE2GP
      4       MG_GP2NE
       5         MG_GENSPHERE
       5         MG_GP2L/PG
       5         MG_GRDZONECNV
     3     MG_UTM2NE
      4       MG_GENSPHERE
      4       MG_NE2GP
       5         MG_GP2NE
        6           MG_GENSPHERE
        6           MG_GP2L/PG
        6           MG_GRDZONECNV
      4       MG_NE2UTM
       5         MG_PG2UPS
        6           MG_NE2UTM
PSA168:Loop detected (see level 4) - Structure truncated.
      4       MG_PG2UPS
       5         MG_NE2UTM
        6           MG_PG2UPS
PSA168:Loop detected (see level 4) - Structure truncated.
      4       MG_UPS2PG
     3     MG_IIINVTL
      4       MG_IITARGLOC
       5         MG_GENSPHERE
      4       MG_GENSPHERE
     3     MG_RMSINVTL
    2   MG_IICOORDCNV
     3     MG_GP2NE
      4       MG_GENSPHERE
      4       MG_GP2L/PG
      4       MG_GRDZONECNV
     3     MG_NE2GP
      4       MG_GP2NE
       5         MG_GENSPHERE
       5         MG_GP2L/PG
       5         MG_GRDZONECNV
     3     MG_UTM2NE
      4       MG_GENSPHERE
      4       MG_NE2GP
   1 MG_Algorithms
    2  MG_BUILDCD
     3    MG_IITARGLOC
      4      MG_GENSPHERE
    2  MG_BUILDSKTCH
     3    MG_IITARGLOC
      4      MG_GENSPHERE
     3    MG_IIFLMITZN
     3    MG_MAPCUR2NE
      4      MG_GP2NE
       5        MG_GENSPHERE
       5        MG_GP2L/PG
       5        MG_GRDZONECNV
      4      MG_GRDZONECNV
      4      MG_NE2GP
       5        MG_GP2NE
        6          MG_GENSPHERE
        6          MG_GP2L/PG
        6          MG_GRDZONECNV
    2  MG_FRAMSRCH
     3    MG_NE2GP
      4      MG_GP2NE
       5        MG_GENSPHERE
       5        MG_GP2L/PG
       5        MG_GRDZONECNV
     3    MG_UTM2NE
      4      MG_GENSPHERE
      4      MG_NE2GP
       5        MG_GP2NE
        6          MG_GENSPHERE
        6          MG_GP2L/PG
        6          MG_GRDZONECNV
      4      MG_NE2UTM
       5        MG_PG2UPS
        6          MG_NE2UTM
PSA168:Loop detected (see level 4) - Structure truncated.
      4      MG_PG2UPS
       5        MG_NE2UTM
        6          MG_PG2UPS
PSA168:Loop detected (see level 4) - Structure truncated.
      4      MG_UPS2PG
     3    MG_IIINVTL
      4      MG_IITARGLOC
       5        MG_GENSPHERE
      4      MG_GENSPHERE
     3    MG_RMSINVTL
    2  MG_IICOORDCNV
     3    MG_GP2NE
      4      MG_GENSPHERE
      4      MG_GP2L/PG
      4      MG_GRDZONECNV
     3    MG_NE2GP
      4      MG_GP2NE
       5        MG_GENSPHERE
       5        MG_GP2L/PG
       5        MG_GRDZONECNV
     3    MG_UTM2NE
      4      MG_GENSPHERE
      4      MG_NE2GP
```

```
         5          MG_GP2NE
          6            MG_GENSPHERE
          6            MG_GP2L/PG
          6            MG_GRDZONECNV
        4         MG_NE2UTM
         5          MG_PG2UPS
          6            MG_NE2UTM
PSA168:Loop detected (see level 4) - Structure truncated.
        4         MG_PG2UPS
         5          MG_NE2UTM
          6            MG_PG2UPS
PSA168:Loop detected (see level 4) - Structure truncated.
        4         MG_UPS2PG
       3        MG_NE2UTM
        4         MG_PG2UPS
         5          MG_NE2UTM
PSA168:Loop detected (see level 3) - Structure truncated.
      2       MG_LORANCNV
       3        MG_GENSPHERE
      2       MG_MAPITZN
       3        MG_LAMBERTCG
      2       MG_NAVCORN
       3        MG_IITARGLOC
        4         MG_GENSPHERE
       3        MG_IIFLMITZN
       3        MG_MAP JR2NE
        4         MG_GP2NE
         5          MG_GENSPHERE
         5          MG_GP2L/PG
         5          MG_GRDZONECNV
        4         MG_GRDZONECNV
        4         MG_NE2GP
         5          MG_GP2NE
          6            MG_GENSPHERE
          6            MG_GP2L/PG
          6            MG_GRDZONECNV
       3        MG_GP2NE
        4         MG_GENSPHERE
        4         MG_GP2L/PG
        4         MG_GRDZONECNV
       3        MG_NE2GP
        4         MG_GP2NE
         5          MG_GENSPHERE
         5          MG_GP2L/PG
         5          MG_GRDZONECNV
       3        MG_UTM2NE
        4         MG_GENSPHERE
        4         MG_NE2GP
         5          MG_GP2NE
          6            MG_GENSPHERE
          6            MG_GP2L/PG
          6            MG_GRDZONECNV
        4         MG_NE2UTM
         5          MG_PG2UPS
          6            MG_NE2UTM
PSA168:Loop detected (see level 4) - Structure truncated.
        4         MG_PG2UPS
         5          MG_NE2UTM
          6            MG_PG2UPS
```

```
PSA168:Loop detected (see level 4) - Structure truncated.
      4       MG_UPS2PG
     3     MG_NE2UTM
      4       MG_PG2UPS
       5         MG_NE2UTM
PSA168:Loop detected (see level 3) - Structure truncated.
    2   MG_PRECPTFLE
     3     MG_RMSTARGLOC
    2   MG_PRIPTDET
     3     MG_IITARGLOC
      4       MG_GENSPHERE
     3     MG_IIFLMITZN
     3     MG_MAPCUR2NE
      4       MG_GP2NE
       5         MG_GENSPHERE
       5         MG_GP2L/PG
       5         MG_GRDZONECNV
      4       MG_GRDZONECNV
      4       MG_NE2GP
       5         MG_GP2NE
        6           MG_GENSPHERE
        6           MG_GP2L/PG
        6           MG_GRDZONECNV
     3     MG_GP2NE
      4       MG_GENSPHERE
      4       MG_GP2L/PG
      4       MG_GRDZONECNV
     3     MG_NE2GP
      4       MG_GP2NE
       5         MG_GENSPHERE
       5         MG_GP2L/PG
       5         MG_GRDZONECNV
     3     MG_UTM2NE
      4       MG_GENSPHERE
      4       MG_NE2GP
       5         MG_GP2NE
        6           MG_GENSPHERE
        6           MG_GP2L/PG
        6           MG_GRDZONECNV
      4       MG_NE2UTM
       5         MG_PG2UPS
        6           MG_NE2UTM
PSA168:Loop detected (see level 4) - Structure truncated.
      4       MG_PG2UPS
       5         MG_NE2UTM
        6           MG_PG2UPS
PSA168:Loop detected (see level 4) - Structure truncated.
      4       MG_UPS2PG
     3     MG_NE2UTM
      4       MG_PG2UPS
       5         MG_NE2UTM
PSA168:Loop detected (see level 3) - Structure truncated.
     3     MG_IIINVTL
      4       MG_IITARGLOC
       5         MG_GENSPHERE
      4       MG_GENSPHERE
     3     MG_RMSINVTL
     3     MG_RMSTARGLOC
     3     MG_NE2MAPCUR
```

```
      4       MG_GP2NE
       5         MG_GENSPHERE
       5         MG_GP2L/PG
       5         MG_GRDZONECNV
      4       MG_GP2L/PG
      4       MG_NE2GP
       5         MG_GP2NE
        6           MG_GENSPHERE
        6           MG_GP2L/PG
        6           MG_GRDZONECNV
    2   MG_PROXSRCH
     3     MG_MAPCUR2NE
      4       MG_GP2NE
       5         MG_GENSPHERE
       5         MG_GP2L/PG
       5         MG_GRDZONECNV
      4       MG_GRDZONECNV
      4       MG_NE2GP
       5         MG_GP2NE
        6           MG_GENSPHERE
        6           MG_GP2L/PG
        6           MG_GRDZONECNV
     3     MG_GP2NE
      4       MG_GENSPHERE
      4       MG_GP2L/PG
      4       MG_GRDZONECNV
     3     MG_UTM2NE
      4       MG_GENSPHERE
      4       MG_NE2GP
       5         MG_GP2NE
        6           MG_GENSPHERE
        6           MG_GP2L/PG
        6           MG_GRDZONECNV
      4       MG_NE2UTM
       5         MG_PG2UPS
        6           MG_NE2UTM
PSA168:Loop detected (see level 4) - Structure truncated.
      4       MG_PG2UPS
       5         MG_NE2UTM
        6           MG_PG2UPS
PSA168:Loop detected (see level 4) - Structure truncated.
      4       MG_UPS2PG
    2   MG_STEROHGHT
     3     MG_IITARGLOC
      4       MG_GENSPHERE
     3     MG_IIFLMITZN
     3     MG_IIINVTL
      4       MG_IITARGLOC
       5         MG_GENSPHERE
      4       MG_GENSPHERE
```

4 MG_DEG2RAD PROCESS

DESCRIPTION:

This algorithm converts angle in degrees/minutes/seconds into its equivalent angle in scaled Pi radians.

SOURCES: CG106100A/Part_I

SECURITY: U

RESP PD: JWG

| ATTRIBUTE: | VALUE: |
|---|---|
| abbreviation | Degrees_2_Binary_{Binary} |
| type_of_source | document |
| date_acquired | '07/01/82' |
| processing_done | none |
| mathematical_field | trigonometry |
| tree_level | leaf |
| requirements_performance | TBD |

6 MG_GENSPHERE PROCESS

DESCRIPTION:

This algorithm determines the appropriate spheroid number corresponding to the input latitude/longitude by scanning a specially constructed map database which contains the relationship between longitude bands/latitude strips and spheroid numbers.

KEYWORDS: algorithm

SOURCES: 3.2.91 CG108100A/Part_I

SECURITY: U

RESP PD: JWG

| ATTRIBUTE: | VALUE: |
|---|---|
| abbreviation | Spheroid_Generation |
| type_of_source | document |
| date_acquired | '07/01/82' |
| processing_done | none |
| mathematical_field | cartography |
| tree_level | leaf |
| requirements_performance | TBD |

```
? MG_GP2L/PG                                                          PROCESS

    DESCRIPTION:
                        This algorithm converts an input geographic coordinate
                   latitude/longitude pair to the equivalent Lambert/polar grid
                   northing/easting coordinate set.

         KEYWORDS: algorithm

          SOURCES: 3.2.115                                CG108100A/Part_I

         SECURITY: U

          RESP PD: JWG

       ATTRIBUTE:                             VALUE:
                   abbreviation                           Lat/Long_2_Lambert/Polar_Grid
                   type_of_source                         document
                   date_acquired                          '07/01/82'
                   processing_done                        none
                   mathematical_field                     cartography
                   tree_level                             leaf
                   requirements_performance               TBD
```

8 MG_GP2NE PROCESS

DESCRIPTION: This algorithm converts and inputs latitude, longitude pair into its equivalent northing and easting coordinate set.

KEYWORDS: algorithm

SOURCES: 3.2.83 CG108100A/Part_I

SECURITY: U

RESP PD: JWG

| ATTRIBUTE: | VALUE: |
|---|---|
| abbreviation | Geographic_2_Northing/Easting |
| type_of_source | document |
| date_acquired | '07/01/82' |
| processing_done | none |
| mathematical_field | cartography |
| tree_level | leaf |
| requirements_performance | TBD |

9 MG_GRDZONECNV PROCESS

DESCRIPTION:

This algorithm converts input longitude and latitude to UTM/UPS Grid Zone number and letter. It fails at the upper (closed) limits of both longitude and latitude.

KEYWORDS: algorithm

SOURCES: 3.2.90 CG108100A/Part_I

SECURITY: U

RESP PD: JWG

| ATTRIBUTE: | VALUE: |
|---|---|
| abbreviation | Grid_Zone_Generation |
| type_of_source | document |
| date_acquired | '07/01/82' |
| processing_done | analyzed/Pascalized |
| mathematical_field | cartography |
| robustness | 3 |
| tree_level | leaf |
| requirements_performance | generally_satisfactory |
| references | '[JWG]DRVGZNG/MGGZDG' |

```
14 MG_L/PG2GP                                                    PROCESS

    DESCRIPTION:
                    This algorithm converts Lambert/polar grid input
                coordinates into equivalent geographic coordinate
                latitude/longitude.

      KEYWORDS: algorithm

       SOURCES: 3.2.116                         CG108100A/Part_I

      SECURITY: U

       RESP PD: JWG

     ATTRIBUTE:                         VALUE:
                abbreviation                 Lambert/Polar_Grid_2_Lat/Long
                type_of_source               document
                date_acquired                '07/01/82'
                processing_done              none
                mathematical_field           cartography
                tree_level                   leaf
                requirements_performance     TBD
```

15 MG_LAMBERTCG PROCESS

DESCRIPTION:

This algorithm calculates the Lambert constants required for use with the Lambert/Polar Grid to UPS and the UPS to Lambert/Polar Grid conversion algorithms.

KEYWORDS: algorithm

SOURCES: 3.2.119 CG108100A/Part_I

SECURITY: U

RESP PD: JWG

| ATTRIBUTE: | VALUE: |
|---|---|
| abbreviation | Lambert_Constant_Generation |
| type_of_source | document |
| date_acquired | '07/01/82' |
| processing_done | analyzed/Pascalized |
| mathematical_field | cartographs |
| robustness | 6 |
| tree_level | leaf |
| requirements_performance | satisfactory |
| references | 'DJWGDLCGPROC |

## 17 MG_MAPCUR2NE PROCESS

DESCRIPTION:

This algorithm converts an xy map cursor position to its equivalent northing/easting or latitude/longitude or grid zone number/letter and spheroid output coordinate.

KEYWORDS: algorithm

SOURCES: 3.2.89 CG108100A/Part_I

SECURITY: U

RESP PD: JWG

| ATTRIBUTE: | VALUE: |
|---|---|
| abbreviation | Map_Cursor_2_Northing_Easting |
| type_of_source | document |
| date_acquired | '07/01/82' |
| processing_done | none |
| mathematical_field | cartography |
| tree_level | middle |
| requirements_performance | TBD |

DESCRIPTION:

This algorithm converts and inputs northing/easting set into a latitude/longitude pair.

KEYWORDS: algorithm

SOURCES: 3.2.84 CG108100A/Part_I

SECURITY: U

RESP PD: JWG

| ATTRIBUTE: | VALUE: |
|---|---|
| abbreviation | Northing/Easting_2_Lat/Long |
| type_of_source | document |
| date_acquired | '07/01/82' |
| processing_done | none |
| mathematical_field | cartography |
| tree_level | leaf |
| requirements_performance | TBD |

01 MG_NE2MAPCUR PROCESS

DESCRIPTION:

This algorithm converts either northing/easting or latitude/longitude or grid zone number/letter pairs into equivalent xy map cursor position pairs for display.

KEYWORDS: algorithm

SOURCES: 3.2.38 CG108100A/Part_I

SECURITY: U

RESP PD: JWG

| ATTRIBUTE: | VALUE: |
|---|---|
| abbreviation | Northing/Easting_2_Map_Cursor |
| type_of_source | document |
| date_acquired | '07/01/32' |
| processing_done | none |
| mathematical_field | cartography |
| tree_level | middle |
| requirements_performance | TBD |

22 MG_NE2UTM PROCESS

DESCRIPTION:

This algorithm converts northing and easting to a composite UTM pair.

KEYWORDS: algorithm

SOURCES: 3.2.86 CG108100A/Part_I

SECURITY: U

RESP PD: JWG

| ATTRIBUTE: | VALUE: |
|---|---|
| abbreviation | Northing/Easting_2_UTM |
| type_of_source | document |
| date_acquired | '07/01/82' |
| processing_done | none |
| mathematical_field | cartography |
| tree_level | leaf |
| requirements_performance | TBD |

3 MG_PG2UPS PROCESS

DESCRIPTION:

This algorithm converts Polar Grid northing/easting coordinates into equivalent Universal Polar Sterographic or Universal Transverse Mercator (utilizing the NE2UTM algorithm).

KEYWORDS: algorithm

SOURCES: 3.2.118 CG108100A/Part_I

SECURITY: U

RESP PD: JWG

| ATTRIBUTE: | VALUE: |
|---|---|
| abbreviation | Polar_Grid_2_UPS |
| type_of_source | document |
| date_acquired | '07/01/82' |
| processing_done | analyzed/Pascalized |
| mathematical_field | cartography |
| robustness | 3 |
| tree_level | leaf |
| requirements_performance | TBD |
| references | '[JWG]PG2UPS' |

27 MG_RAD2DEG PROCESS

DESCRIPTION:

This algorithm converts an angle in scaled Pi radians into an equivalent angle in degrees/minutes/seconds.

SOURCES: CG108100A/Part_I

SECURITY: U

RESP PD: JWG

| ATTRIBUTE: | VALUE: |
|---|---|
| abbreviation | Radians_2_Degrees |
| type_of_source | document |
| date_acquired | '07/01/82' |
| processing_done | none |
| mathematical_field | trigonometry |
| tree_level | leaf |
| requirements_performance | TBD |

33 MG_UPS2PG PROCESS

DESCRIPTION:

This algorithm converts Universal Polar Sterographic into equivalent polar grid coordinates.

KEYWORDS: algorithm

SOURCES: 3.2.117 CG108100A/Part_I

SECURITY: U

RESP PD: JWG

| ATTRIBUTE: | VALUE: |
|---|---|
| abbreviation | UPS_2_Polar_Grid |
| type_of_source | document |
| date_acquired | '07/01/82' |
| processing_done | none |
| mathematical_field | cartography |
| tree_level | leaf |
| requirements_performance | TBD |

34 MG_UTM2NE PROCESS

DESCRIPTION:

This algorithm converts a UTM coordinate set into the equivalent composite northing and easting pair.

KEYWORDS: algorithm

SOURCES: 3.2.85 CG108100A/Part_I

SECURITY: U

RESP PD: JWG

| ATTRIBUTE: | VALUE: |
|---|---|
| abbreviation | UTM_2_Northing/Easting |
| type_of_source | document |
| date_acquired | '07/01/82' |
| processing_done | none |
| mathematical_field | cartography |
| tree_level | leaf |
| requirements_performance | TBD |

### 7.3. Algorithms in Standard Form

```
PROGRAM DriverLambertConstantGeneration (INPUT,OUTPUT);
{                                                                 }
{ DriverLambertConstantGeneration provides an environment for     }
{ DriverLambertConstantGeneration PROCEDURE                       }
{ testing the LambertConstantGeneraton procedure                  }
{                                                                 }
{ Interfaces with the LambertConstantGeneration procedure are:    }
{   GLOBAL VARIABLES    -    Eccentricity                         }
{                            SemimajorAxis                        }
{   INPUT PARAMETERS    -    LambdaL                              }
{                            LambdaR                              }
{                            PhiU                                 }
{                            PhiL                                 }
{                            Phi1                                 }
{                            Phi2                                 }
{   OUTPUT PARAMETERS   -    Kappa                                }
{                            Iota                                 }
{                            ProjectionConeRadius                 }
{                            LambdaC                              }
{                            SquaredEccentricity                  }
{                            Hemisphere                           }
{                                                                 }
{ declarations                                                    }
{                                                                 }
TYPE
        PiRadians                   = REAL;
        ZeroToOne                   = REAL;
        Flag                        = (northern,southern);
{                                                                 }
VAR
        Eccentricity                : ZeroToOne;
        SemimajorAxis               : REAL;

        LambdaL                     : PiRadians;
        LambdaR                     : PiRadians;
        PhiU                        : PiRadians;
        PhiL                        : PiRadians;
        Phi1                        : PiRadians;
        Phi2                        : PiRadians;
{                                                                 }
        Kappa                       : REAL;
        Iota                        : REAL;
        ProjectionConeRadius        : REAL;
        LambdaC                     : PiRadians;
        SquaredEccentricity         : ZeroToOne;
        Hemisphere                  : Flag;
{                                                                 }
{                                                                 }
PROCEDURE LambertConstantGeneration
           ( {GLOBAL}  Eccentricity             : ZeroToOne;
             {GLOBAL}  SemimajorAxis            : REAL;
             {IN}      LambdaL                  : PiRadians;
             {IN}      LambdaR                  : PiRadians;
             {IN}      PhiU                     : PiRadians;
             {IN}      PhiL                     : PiRadians;
             {IN}      Phi1                     : PiRadians;
             {IN}      Phi2                     : PiRadians;
             {OUT} VAR Kappa                    : REAL;
             {OUT} VAR Iota                     : REAL;
             {OUT} VAR ProjectionConeRadius     : REAL;
             {OUT} VAR LambdaC                  : PiRadians;
             {OUT} VAR SquaredEccentricity      : ZeroToOne;
             {OUT} VAR Hemisphere               : Flag    );EXTERN;
```

```
{                                                                   }
{ Procedure LambertConstantGeneration models the Lambert Constant }
{ Generation algorithm described in paragraph 3.2.119 of document }
{ CG108100a dated 23 October 1978.                                  }
{                                                                   }
   { J.W.Gillis 4-22-82                                             }
{                                                                   }
   { This procedure ASSUMES that certain data are available as      }
   { required by the algorithm but not described as inputs in the }
   { reference document. These data are: Eccentricity               }
   {                                      SemimajorAxis             }
{                                                                   }
{                                                                   }
   { This procedure DOESNOT perform data validation checks that are   }
   { not specified in the algorithm description. This is to allow the }
   { algorithm features to be presented more clearly.                 }
{                                                                     }
{ executables                                                         }
{                                                                     }
BEGIN
{                                                                     }
        { establish GLOBAL variables}
{                                                                     }
        Eccentricity              := 0.5;
        SemimajorAxis             := 10.0;
{                                                                     }
        { establish INPUT PARAMETERS}
{                                                                     }
        LambdaL                   := 0.25;
        LambdaR                   := 0.0;
        PhiU                      := 0.125;
        PhiL                      := 0.0;
        Phi1                      := 0.0675;
        Phi2                      := 0.375;
{                                                                     }
{ echo INPUT PARAMETERS                                               }
{                                                                     }
        WRITELN;
        WRITELN (' setup Eccentricity is ',Eccentricity);
        WRITELN (' setup SemimajorAxis is ',SemimajorAxis);
        WRITELN (' setup LambdaL is ',LambdaL);
        WRITELN (' setup LambdaR is ',LambdaR);
        WRITELN (' setup PhiU is ',PhiU);
        WRITELN (' setup PhiL is ',PhiL);
        WRITELN (' setup Phi1 is ',Phi1);
        WRITELN (' setup Phi2 is ',Phi2);
{                                                                     }
        LambertConstantGeneration ( {GLOBAL}  Eccentricity,
                                    {GLOBAL}  SemimajorAxis,
                                    {OUT}     LambdaL,
                                    {OUT}     LambdaR,
                                    {OUT}     PhiU,
                                    {OUT}     PhiL,
                                    {OUT}     Phi1,
                                    {OUT}     Phi2,
                                    {IN}      Kappa,
                                    {IN}      Iota,
                                    {IN}      ProjectionConeRadius,
                                    {IN}      LambdaC,
                                    {IN}      SquaredEccentricity,
                                    {IN}      Hemisphere );
{                                                                     }
        { list output parameters from log}
{                                                                                }
          WRITELN;
          WRITELN (' Kappa is ' ,Kappa);
          WRITELN (' Iota is ' ,Iota);
          WRITELN (' ProjectionConeRadius is ',ProjectionConeRadius);
          WRITELN (' LambdaC is ',LambdaC);
          WRITELN (' SquaredEccentricity is ',SquaredEccentricity);
          WRITELN (' Hemisphere is ',Hemisphere)
{                                                                                }
END. { DriverLambertConstantGeneration PROCEDURE                                 }
```

```
MODULE LedProc (INPUT,OUTPUT);
{                                                                         }
{
{
TYPE
          zeroToOne                                    = REAL;
          PiRadians                                    = REAL;
          Flag                                         = (northern,southern);
{                                                                         }
{
{
PROCEDURE LambertConstantGeneration
          ( {GLOBAL}  Eccentricity                     : ZeroToOne;
            {GLOBAL}  SemimajorAxis                    : REAL;
            {IN}      LambdaL                          : PiRadians;
            {IN}      LambdaR                          : PiRadians;
            {IN}      PhiU                             : PiRadians;
            {IN}      PhiL                             : PiRadians;
            {IN}      Phi1                             : PiRadians;
            {IN}      Phi2                             : PiRadians;
            {OUT} VAR Kappa                            : REAL;
            {OUT} VAR Iota                             : REAL;
            {OUT} VAR ProjectionConeRadius             : REAL;
            {OUT} VAR LambdaC                          : PiRadians;
            {OUT} VAR SquaredEccentricity              : ZeroToOne;
            {OUT} VAR Hemisphere                       : Flag    );
{                                                                         }
{ Generation algorithm described in paragraph 3.2.119 of document
{ CG108100a dated 23 October 1978.                                        }
{                                                                         }
{ J.W.Gillis 4-27-82
{
{ This procedure ASSUMES that certain data are available as               }
{ required by the algorithm but not described as inputs in the            }
{ reference document. These data are: Eccentricity
{                                     SemiimajorAxis                      }
{                                                                         }
{ This procedure DOESNOT perform data validation checks that are
{ not specified in the algorithm description. This is to allow the
{ algorithm features to be presented more clearly.                        }
{                                                                         }
{
{                                                                         }
CONST
          PiOver2                                      = 1.57079;
{
VAR
          Phi                                          : ARRAY [1..3] OF PiRadians;
          PhiPrime                                     : ARRAY [1..3] OF PiRadians;
          Zeta                                         : ARRAY [1..3] OF PiRadians;
          Curvature                                    : ARRAY [1..2] OF REAL;
          Index                                        :INTEGER;
BEGIN
{                                                                         }
   Phi[1] := Phi1;
   Phi[2] := Phi2;
   Phi[3] := (PhiU+PhiL)/2.0;
{                                                                         }
   LambdaC := (LambdaL + LambdaR)/2.0;
{
   SquaredEccentricity := SQR(Eccentricity);
{                                                                         }
   FOR Index :=1 TO 3 DO
```

```
      PhiPrime[Index] := ARCTAN((1.0-SquaredEccentricity)*
                                 (SIN(ABS(Phi[Index]))/
                                  COS(ABS(Phi[Index]))));
{                                                                        }
   FOR Index := 1 TO 2 DO
      Curvature[Index] := SemimajorAxis/
                          (SQRT(1.0-SquaredEccentricity*
                                SQR(SIN(ABS(Phi[Index])))));
   FOR Index := 1 TO 3 DO
      Zeta[Index] := PiOver2-PhiPrime[Index];
{                                                                        }
   Iota := (LN(COS(ABS(Phi[1])))-LN(COS(ABS(Phi[2])))+
            LN(Curvature[1])-LN(Curvature[2]))/
           (LN(SIN(Zeta[1]/2.0)/COS(Zeta[1]/2.0))-
            LN(SIN(Zeta[2]/2.0)/COS(Zeta[2]/2.0)));
{                                                                        }
   Kappa := (Curvature[1]*COS(ABS(Phi[1])))/
            (Iota*((SIN(Zeta[1]/2.0)/COS(Zeta[1]/2.0))**Iota));
{                                                                        }
   ProjectionConeRadius := Kappa*((SIN(Zeta[3]/2.0)/
                                  COS(Zeta[3]/2.0))**Iota);
{                                                                        }
   { NO ALGORITHM is available for the Hemisphere Flag set }
{                                                                        }
   { list intermediate values }
{                                                                        }
   WRITELN;
   WRITELN (' computed Phi[3] is ',Phi[3]);
   FOR Index := 1 TO 3 DO
      WRITELN (' computed PhiPrime[Index] is ',PhiPrime[Index]);
   FOR Index := 1 TO 2 DO
      WRITELN (' computed Curvature[Index] is ',Curvature[Index]);
   FOR Index := 1 TO 3 DO
      WRITELN (' computed Zeta[Index] is ',Zeta[Index]);
{                                                                        }
END; { LambertConstantGeneration PROCEDURE }
{                                                                        }
END. { LcgProc MODULE }
```

```
{                                                                                                  }
PROGRAM Drvit (INPUT,OUTPUT);
{                                                                                                  }
{                                                                                                  }
TYPE
         Meters                      =REAL;
         Decameters                  =REAL;
         Letter                      ='A'..'Z';
         Spheres                     =INTEGER;
{                                                                                                  }
VAR    PGNorthingCoord          :Meters;
       PGEastingCoord           :Meters;
       PGZoneLetter             :Letter;
       PGZoneNumber             :INTEGER;
       SpheroidNumber           :Spheres;
       UPSZoneLetter            :Letter;
       UPSEastingLetter         :Letter;
       UPSNorthingLetter        :Letter;
       UPSEastingNumber         :Decameters;
       UPSNorthingNumber        :Decameters;
{                                                                                                  }
{                                                                                                  }
PROCEDURE PolarToUPS
            ({IN }      PGNorthingCoord          :Meters;
             {IN }      PGEastingCoord           :Meters;
             {IN }      PGZoneLetter             :Letter;
             {IN }      PGZoneNumber             :INTEGER;
             {IN }      SpheroidNumber           :Spheres;
             {OUT} VAR UPSZoneLetter            :Letter;
             {OUT} VAR UPSEastingLetter         :Letter;
             {OUT} VAR UPSNorthingLetter        :Letter;
             {OUT} VAR UPSEastingNumber         :Decameters;
             {OUT} VAR UPSNorthingNumber        :Decameters);EXTERN;
{                                                                                                  }
    BEGIN
{                                                                                                  }
WRITELN;
WRITELN (' ENTER PG NORTHING COORD ');
READLN (PGNORTHINGCOORD);
WRITELN;
WRITELN (' ENTER PG EASTING COORD ');
READLN (PGEASTINGCOORD);
WRITELN;
WRITELN (' ENTER PG ZONE NUMBER ');
READLN (PGZONENUMBER);
WRITELN;
WRITELN (' ENTER PG ZONE LETTER ');
READLN (PGZONELETTER);
WRITELN;
WRITELN (' ENTER SPHEROID NUMBER ');READLN (SPHEROIDNUMBER);
{                                                                                                  }
{                                                                                                  }
PolarToUPS (PGNorthingCoord,
              PGEastingCoord,
              PGZoneLetter,
              PGZoneNumber,
              SpheroidNumber,
              UPSZoneLetter,
              UPSEastingLetter,
              UPSNorthingLetter,
              UPSEastingNumber,
              UPSNorthingNumber);
{                                                                                                  }
```

```
6400   END. {of PROGRAM Drvit}
```

```
{                                                                      }
MODULE PG2UPS (INPUT,OUTPUT);
{                                                                      }
{                                                                      }
TYPE
        Meters                  =REAL;
        Decameters              =REAL;
        Letter                  ='A'..'Z';
        Spheres                 =INTEGER;
{                                                                      }
{                                                                      }
{                                                                      }
PROCEDURE PolarToUPS
           ({IN }     PGNorthingCoord       :Meters;
            {IN }     PGEastingCoord        :Meters;
            {IN }     PGZoneLetter          :Letter;
            {IN }     PGZoneNumber          :INTEGER;
            {IN }     SpheroidNumber        :Spheres;
            {OUT} VAR UPSZoneLetter         :Letter;
            {OUT} VAR UPSEastingLetter      :Letter;
            {OUT} VAR UPSNorthingLetter     :Letter;
            {OUT} VAR UPSEastingNumber      :Decameters;
            {OUT} VAR UPSNorthingNumber     :Decameters);
{                                                                      }
{                                                                      }
{ PROCEDURE PolarToUPS models the Polar Grid to Universal Polar        }
{ Steriographic conversion algorithm described in paragraph            }
{ 3.2.118 of CG108100A dated 23 October 1978.                          }
{                                                                      }
{ PROCEDURE PolarToUPS CALLs: PROCEDURE ConvertToUPS                   }
{                             PROCEDURE ConvertToUTM                   }
{                                                                      }
{ Programmed by J.W.gillis       5-5-82                                }
{                                                                      }
{ This procedure ASSUMES that certain data are available as            }
{ required by the algorithm but not adequately described in the        }
{ algorithm description.                                               }
{                                                                      }
{ This procedure DOESNOT perform data validity checks that are         }
{ not specified in the algorithm description. This is to allow         }
{ the algorithm features to be presentes more clearly.                 }
{                                                                      }
{ PROCEDURE PolarToUPS accepts northing and easting coordinates,       }
{ tests whether the UPS or UTM grid system is a suitable target        }
{ system. If the UPS grid system is appropriate, the the               }
{ conversion is performed by this procedure. If the UTM grid           }
{ system is appropriate, then this procedure CALLs the                 }
{ PolarToUTM procedure (algorithm 3.2.86) to perform the               }
{ conversion.                                                          }
{                                                                      }
{                                                                      }
{
TYPE
       Meters                 = REAL;
       Decameters             = REAL;
       Letter                 = 'A'..'Z';
       Spheres                = INTEGER;
{                                                                      }
VAR
       UTMZoneLetter        :Letter;
       UTMEastingLetter     :Letter;
       UTMNorthingLetter    :Letter;
       UTMEastingNumber     :Decameters;
```

```
      UTMNorthingNumber   :Decameters;
{                                                                }
PROCEDURE ConvertToUPS
          ({IN }     PGNorthingCoord     :Meters;
           {IN }     PGEastingCoord      :Meters;
           {IN }     PGZoneLetter        :letter;
           {IN }     PGZoneNumber        :INTEGER;
           {OUT} VAR UPSZoneLetter       :Letter;
           {OUT} VAR UPSEastingLetter    :Letter;
           {OUT} VAR UPSNorthingLetter   :Letter;
           {OUT} VAR UPSEastingNumber    :Decameters;
           {OUT} VAR UPSNorthingNumber   :Decameters);EXTERN;
{                                                                }
PROCEDURE ConvertToUTM
          ({IN }     PGNorthingCoord     :Meters;
           {IN }     PGEastingCoord      :Meters;
           {IN }     PGZoneLetter        :Letter;
           {IN }     PGZoneNumber        :INTEGER;
           {IN }     SpheroidNumber      :Spheres;
           {OUT} VAR UTMZoneLetter       :Letter;
           {OUT} VAR UTMEastingLetter    :Letter;
           {OUT} VAR UTMNorthingLetter   :Letter;
           {OUT} VAR UTMEastingNumber    :Decameters;
           {OUT} VAR UTMNorthingNumber   :Decameters);EXTERN;
{                                                                }
{                                                                }
   {Select the appropriate grid reference system                 }
   {                                                             }
   BEGIN
WRITELN (' PG2UPS ALL HOOKED UP');
      IF PGZoneNumber = 0
         THEN ConvertToUPS ({OUT} PGNorthingCoord,
                            {OUT} PGEastingCourd,
                            {OUT} PGZoneLetter,
                            {OUT} PGZoneNumber,
                            {IN } UPSZoneLetter,
                            {IN } UPSEastingLetter,
                            {IN } UPSNorthingLetter,
                            {IN } UPSEastingNumber,
                            {IN } UPSNorthingNumber)
         ELSE ConvertToUTM ({OUT} PGNorthingCoord,
                            {OUT} PGEastingCoord,
                            {OUT} PGZoneLetter,
                            {OUT} PGZoneNumber,
                            {OUT} SpheroidNumber,
                            {IN } UTMZoneLetter,
                            {IN } UTMEastingLetter,
                            {IN } UTMNorthingLetter,
                            {IN } UTMEastingNumber,
                            {IN } UTMNorthingNumber);
{                                                                }
{                                                                }
END; {of PROCEDURE PolarToUPS}
{                                                                }
{                                                                }
END. {of MODULE PG2UPS}
```

```
{                                                                        }
MODULE CNV2UPS (INPUT,OUTPUT);
{                                                                        }
{                                                                        }
TYPE
        Meters                 =REAL;
        Decameters             =REAL;
        Letter                 ='A'..'Z';
{                                                                        }
PROCEDURE ConvertToUPS
           ({IN }     PGNorthingCoord     :Meters;
            {IN }     PGEastingCoord      :Meters;
            {IN }     PGZoneLetter        :Letter;
            {IN }     PGZoneNumber        :INTEGER;
            {OUT} VAR UPSZoneLetter       :Letter;
            {OUT} VAR UPSEastingLetter    :Letter;
            {OUT} VAR UPSNorthingLetter   :Letter;
            {OUT} VAR UPSEastingNumber    :Decameters;
            {OUT} VAR UPSNorthingNumber   :Decameters);
{                                                                        }
{                                                                        }
{ PROCEDURE ConvertToUPS performs the Polar Grid to Universal            }
{ Polar Steriographic conversion algorithm described in                  }
{ paragraph 3.2.118 of CG108100A dated 23 October 1978.                  }
{                                                                        }
{ PROCEDURE ConvertToUPS is CALLed by: PROCEDURE PolarToUPS              }
{                                      PROCEDURE PolarToUTM              }
{                                                                        }
{ Programmed by J.w.gillis       5-6-82                                  }
{                                                                        }
{ This procedure ASSUMES that certain data are available as              }
{ required by the algorithm but not adequately described in the          }
{ algorithm description.                                                 }
{                                                                        }
{ This procedure DOESNOT perform data validity checks that are           }
{ not specified in the algorithm description. This is to allow           }
{ the algorithm features to be presented more clearly.                   }
{                                                                        }
{ PROCEDURE ConvertToUPS accepts polar grid northing and easting}
{ coordinates and converts them to UPS coordinates.                      }
{                                                                        }
{                                                                        }
TYPE
       Meters                 = REAL;
       Decameters             = REAL;
       Letter                 = 'A'..'Z';
{                                                                        }
VAR
       Index                  : INTEGER;
       GridLetter             : ARRAY [1..26] OF Letter     ;
{                                                                        }
FUNCTION aMODb ({IN} a,b:REAL):INTEGER;
   {                                                                     }
   VAR Ainteger               : INTEGER;
         Binteger             : INTEGER;
   {                                                                     }
   BEGIN
       Ainteger:=TRUNC(a);
       Binteger:=TRUNC(b);
       aMODb:=Ainteger MOD Binteger
END; {of FUNCTION MODab}
{                                                                        }
{                                                                        }
{                                                               }
   BEGIN
   {                                                            }
   {                                                            }
      {Initialize the GridLetter array.                  }
      {                                                         }
      FOR Index := 1 TO 26 DO
          GridLetter[Index]:=CHR(ORD('A')+Index-1);
      {                                                         }
      {                                                         }
      {                                                         }
      {Calculate the UPSEastingNumber & UPSEastingLetter Index  }
      {                                                         }
      IF PGZoneLetter > 'M'
         THEN BEGIN
                 UPSEastingNumber:=aMODb((PGEastingCoord-200000),
                                         100000)/10;
                 Index:=aMODb((PGEastingCoord-200000)/1000000,20)
      END; {of IF}
      {                                                         }
      IF PGZoneLetter < 'N'
         THEN BEGIN
                 UPSEastingNumber:=aMODb(PGEastingCoord,
                                         100000)/10;
                 Index:=aMODb(PGEastingCoord/100000,18)
      END; {of IF}
      {                                                         }
WRITELN;
WRITELN ('UPSEASTINGNUMBER IS ',UPSEASTINGNUMBER);
WRITELN ('AND ITS LETTER INDEX IS ',INDEX);
      {                                                         }
      {Calculate the UPSEastingLetter from its index.           }
      {                                                         }
      IF Index > 16
         THEN Index:=Index+2;
      IF (Index > 2) AND (Index < 17)
         THEN Index:=Index+1;
      UPSEastingLetter:=GridLetter[Index];
      {                                                         }
WRITELN;
WRITELN ('UPSEASTINGLETTER IS ',UPSEASTINGLETTER);
WRITELN ('AND ITS CORRECTED INDEX IS ',INDEX);
      {                                                         }
      {                                                         }
      {Calculate the UPSNorthingNumber & NorthingLetter Index   }
      {                                                         }
      IF PGZoneLetter > 'M'
         THEN BEGIN
                 UPSNorthingNumber:=aMODb((PGNorthingCoord-1300000),
                                          100000)/10;
                 Index:=aMODb(((PGNorthingCoord-1300000)/100000),
                              24)
      END; {of IF}
      {                                                         }
      IF PGZoneLetter < 'N'
         THEN BEGIN
                 UPSNorthingNumber:=aMODb((PGNorthingCoord-800000),
                                          100000)/10;
                 Index:=aMODb(((PGNorthingCoord-800000)/100000),
                              24)
      END; {of IF}
      {                                                         }
      {                                                         }
```

```
WRITELN;
WRITELN ('UPSNORTHINGNUMBER IS ',UPSNORTHINGNUMBER);
WRITELN ('AND ITS LETTER INDEX IS ',INDEX);
      {Calculate the UPSNorthingLetter from its Index                }
      {                                                              }
      IF Index > 16
         THEN Index:=Index+2;
      IF Index < 17
         THEN INDEX:=Index+1;
      UPSNorthingLetter:=GridLetter[Index];
      {                                                              }
WRITELN;
WRITELN ('UPSNORTHINGLETTER IS ',UPSNORTHINGLETTER);
WRITELN ('AND ITS CORRECTED INDEX IS ',INDEX);
      {                                                              }
      {                                                              }
{                                                                    }
{                                                                    }
END; {of PROCEDURE ConvertToUPS}
{                                                                    }
END. {of MODULE CNV2UPS}
```

TEMPORARY

STUB PROCEDURE

```
{                                                                    }
MODULE CNV2UTM (INPUT,OUTPUT);
{                                                                    }
{                                                                    }
TYPE
        Meters                 =REAL;
        Decameters             =REAL;
        Letter                 =SET OF CHAR;
        Spheres                =INTEGER;
{                                                                    }
{                                                                    }
PROCEDURE ConvertToUTM
             ({IN }     PGNorthingCoord      :Meters;
              {IN }     PGEastingCoord       :Meters;
              {IN }     PGZoneLetter         :Letter;
              {IN }     PGZoneNumber         :INTEGER;
              {IN }     SpheroidNumber       :Spheres;
              {OUT} VAR UTMZoneLetter        :Letter;
              {OUT} VAR UTMEastingLetter     :Letter;
              {OUT} VAR UTMNorthingLetter    :Letter;
              {OUT} VAR UTMEastingNumber     :Decameters;
              {OUT} VAR UTMNorthingNumber    :Decameters);
{                                                                    }
{                                                                    }
{ PROCEDURE ConvertToUTM performs the Polar Grid to Universal        }
{ Transverse Mercator conversion algorithm described in              }
{ paragraph 3.2.86 of CG108100A dated 23 October 1978.              }
{                                                                    }
{ PROCEDURE ConvertToUTM is CALLed by: PROCEDURE PolarToUTM          }
{                                      PROCEDURE PolarToUPS          }
{                                                                    }
{ Programmed by J.w.gillis       5-6-82                              }
{                                                                    }
{ This procedure ASSUMES that certain data are available as          }
{ required by the algorithm but not adequately described in the      }
{ algorithm description.                                             }
{                                                                    }
{ This procedure DOESNOT perform data validity checks that are       }
{ not specified in the algorithm description. This is to allow       }
{ the algorithm features to be presentes more clearly.               }
{                                                                    }
{ PROCEDURE ConvertToUTM accepts polar grid northing and easting}
{ coordinates and converts them to UTM coordinates.                  }
{                                                                    }
{                                                                    }
TYPE
       Meters                  = REAL;
       Decameters              = REAL;
       Letter                  = SET OF CHAR;
       Spheres                 = INTEGER;
{                                                                    }
{                                                                    }
   BEGIN
{      IF      }
          {THEN}
WRITELN (' CNV2UTM HOOKED UP OK')
{                                                                    }
{                                                                    }
END; {of PROCEDURE ConvertToUTM}
{                                                                    }
END. {of MODULE Cnv2UTM}
```

```
{}
 PROGRAM ConvertToUTM (INPUT,OUTPUT);
 {}
 TYPE
   ASCIIArray         = ARRAY [1..4] OF CHAR;
    Letter            = CHAR;
{}
VAR
    UTMCoordinate          : RECORD
       UTMZoneNumber      :ARRAY[1..2] OF INTEGER;
       UTMZoneLetter      :Letter;
       UTMEastingLetter   :Letter;
       UTMNorthingLetter  :Letter;
       UTMEastingNumber   :ARRAY[1..4]  OF CHAR;
       UTMNorthingNumber  :ARRAY[1..4]  OF CHAR
    END;  {UTMC }
{}
    PGNorthingCoord    :INTEGER; {IN}
    PGEastingCoord     :INTEGER; {IN}
    PGZoneLetter       :Letter ; {IN}
    PGZoneNumber       :INTEGER; {IN}
    SpheroidNumber     :INTEGER; {IN}
    I, EletterOffset, Tel, Tp, Tnl: INTEGER; {Local}
    Number               : INTEGER;
{}
{}
{ PROCEDURE ConvertToUTM performs the Polar Grid to Universal    }
{ Transverse Mercator conversion algorithm described in          }
{ paragraph 3.2.86 of CG108100A dated 23 October 1978.           }
{}
{ PROCEDURE ConvertToUTM is CALLed by: PROCEDURE PolarToUTM      }
{                                      PROCEDURE PolarToUPS      }
{}
{}
{ This procedure ASSUMES that certain data are available as      }
{ required by the algorithm but not adequately described in the }
{ algorithm description.                                         }
{}
{ This procedure DOESNOT perform data validity checks that are  }
{ not specified in the algorithm description. This is to allow  }
{ the algorithm features to be presentes more clearly.          }
{}
{ PROCEDURE ConvertToUTM accepts polar grid northing and easting}
{ coordinates and converts them to UTM coordinates.             }
{}
{}
  PROCEDURE ConvertToASCII( Number: INTEGER {in};
                            VAR AlphaNum:ASCIIArray );
{}
    { Convert an integer number to its ASCII representation in base 10 }
    { This procedure not detailed in source }
{}
    VAR   I: INTEGER;
        Adj: INTEGER;
{}
    BEGIN
      FOR I := 4 DOWNTO 1 DO
          BEGIN
            Adj := (Number MOD 10) + ORD('0') ;
            AlphaNum[I] := CHR(Adj) ;
             Number := Number DIV 10
          END
{}
```

```
 END; {ConvertToASCII}
{}
{}
BEGIN {ConvertToUTM}
{}
{ read input }
{}
WRITELN;
WRITELN ('PG Northing Coord (integer) ') ;
READLN (PGNorthingCoord) ;
WRITELN ('PG Easting Coord (integer) ') ;
READLN (PGEastingCoord) ;
WRITELN ('PG Zone Letter (Letter) ') ;
READLN (PGZoneLetter) ;
WRITELN ('PG Zone Number (integer) ') ;
READLN (PGZoneNumber) ;
WRITELN ('Spheroid Number (integer) ') ;
READLN (SpheroidNumber) ;
{}
{ end of input }
{}
    WITH UTMCoordinate DO
       BEGIN
     { Find Easting Letter (A2) }
      I := PGZoneNumber MOD 3 ;
     { Easting Letter Offset Factor }
     CASE I OF
        0: ELetterOffset := 24;
        1: ELetterOffset :=  6;
        2: ELetterOffset := 15
     END;
{}
     Tel := ( PGEastingCoord - 500000 ) DIV 100000 + ELetterOffset;
{}
{ Following letter conversion not according to documentation; }
{ Procedure in documentation fails at least at Bad Hersfeld, FRG,    }
{ which is in 32U, and this one succeeds at least there.         }
{}
     IF Tel > 15 THEN Tel := Tel + 2
        ELSE IF Tel > 10 THEN Tel := Tel - 1
        ELSE Tel := Tel - 2 ;
      Tel := Tel + ORD('A') - 1 ;
     UTMEastingLetter := Chr( Tel ) {A2};
{}
     { Find Northing Number }
IF NOT ODD( PGZoneNumber) THEN
   PGNorthingCoord := PGNorthingCoord + 500000;
Tp := PGNorthingCoord MOD 2000000;
Tnl := Tp DIV 100000 + 1;
{ Make Spheroid Adjustment }
CASE SpheroidNumber OF
{}
    { Clark 1866 }
    1: IF (PGZoneNumber< 31) OR (PGZoneNumber > 58) THEN Tnl := Tnl + 10
       ELSE IF (PGZoneNumber>51) AND (PGZoneNumber<59) {typo in source}
                  THEN Tnl := Tnl -10;
    { International }
    2: IF (PGZoneNUMBER >46) AND (PGZoneNumber <52)
           THEN Tnl := Tnl + 10;
{}
    { Clark 1880 }
    3: Tnl := Tnl + 10;
{}
```

```
   { Everest }
   4: IF FGZoneNumber < 46 THEN Tnl := Tnl + 10;
{}
   { Bessel }
   5: IF (FGZoneNumber<32) AND (FGZoneLetter>'R') THEN Tnl :- Tnl + 10;
{}
   { Australian }
   6: IF ODD( FGZoneNumber ) THEN Tnl := Tnl   15
         ELSE Tnl := Tnl + 5
{}
END; {CASE OF SpheroidNumber}
{}
{ Calculate Final Northing Letter Index }
{}
IF Tnl > 14 THEN Tnl := Tnl + 2
   ELSE IF Tnl > 9 THEN Tnl := Tnl + 1;
Tnl := Tnl + ORD('A') - 1 ;
UTMNorthingLetter := CHR( Tnl );
{ Format for Output }
ConvertToASCII( FGNorthingCoord MOD 100000, UTMNorthingNumber );
ConvertToASCII( FGEastingCoord MOD 100000, UTMEastingNumber );
UTMZoneNumber[1] := FGZoneNumber DIV 10;
UTMZoneNumber[2] := FGZoneNumber MOD 10 ;
UTMZoneLetter := FGZoneLetter ;
{}
{ WRITE OUTPUT }
{}
WRITELN ;
WRITELN ;
WRITELN;
FOR I := 1 TO 2 DO WRITELN ('UTM Zone Number ',I,UTMZoneNumber[I]);
WRITELN ('UTM Zone Letter            ',UTMZoneLetter) ;
WRITELN ('UTM Easting Letter         ',UTMEastingLetter) ;
WRITELN ('UTM Northing Letter        ',UTMNorthingLetter) ;
FOR I := 1 TO 4 DO BEGIN
WRITELN ('UTM Easting Number  ',I,'       ',UTMEastingNumber[I]);
WRITELN ('UTM Northing Number ',I,'       ',UTMNorthingNumber[I]);
END ;
{}
{ END OF OUTPUT }
{}
END {of WITH UTM }
{}
END. {ConvertToUTM}
```

```
PROCEDURE ConvertToUTM IS
--
TYPE ShortArray IS ARRAY(1..4) OF CHARACTER;
--
   PGNorthingCoord    :INTEGER; --IN
   PGEastingCoord     :INTEGER; --IN
   PGZoneLetter       :CHARACTER; --IN
   PGZoneNumber       :INTEGER; --IN
   SpheroidNumber     :INTEGER; --IN
   --
   UTMZoneNumber      :ARRAY(1..2) OF INTEGER;
   UTMZoneLetter      :CHARACTER;
   UTMEastingLetter   :CHARACTER;
   UTMNorthingLetter  :CHARACTER;
   UTMEastingNumber   :ARRAY(1..4) OF CHARACTER;
   UTMNorthingNumber  :ARRAY(1..4) OF CHARACTER;
   --
   I, EletterOffset, Tel, Tp, tnl, ASCIIPos: INTEGER; --Local
--
 PROCEDURE ConvertToASCII( Number: in INTEGER;
                        AlphaNum: out ShortArray) IS
   -- Convert an integer number to its ASCII representation in base 10
   -- This procedure not detailed in source
   BEGIN
     FOR I in reverse 1 .. 4 LOOP
            ASCIIPos := Number MOD 10 + CHARACTER'POS('0');
            AlphaNum(i) := CHARACTER'VAL( ASCIIPos );
            Number := Number / 10;
          END LOOP;
END ConvertToASCII;
FUNCTION ODD( I: INTEGER ) return BOOLEAN is
BEGIN
   IF I = 2 * ( I/2 ) THEN return false;
   Else return true;
   END IF;
END ODD;
BEGIN
   -- Find Easting Letter (A2)
   I := PGZoneNumber MOD 3;
   -- Easting Letter Offet Factor
   CASE I IS
      WHEN 0=>ELetterOffet := 24;
      WHEN 1=>ELetterOffet :=  6;
      WHEN 2=>ELetterOffset := 15;
      WHEN OTHERS => NULL;
   END CASE;
   --
   Tel := ( PGEastingCoord - 500000 ) / 100000 + ELetterOffset;
   IF Tel > 14 THEN Tel := Tel + 2;
      ELSIF Tel > 9 THEN Tel := Tel + 1;
   END IF;
   Tel := Tel + CHARACTER'POS('a') - 1;
   UTMEastingLetter := CHARACTER'VAL( Tel ); --A2
   --
   -- Find Northing Number
   IF NOT Odd( PGZoneNumber ) THEN
      PGNorthingCoord := PGNorthingCoord + 500000;
   END IF;
```

```
    Tp := PGNorthingCoord MOD 2000000;
    Tnl  := Tp / 100000 + 1;
    -- Make Spheroid Adjustment
    CASE SpheroidNumber IS
       -- Clark 1866
       WHEN 1=>IF PGGridzoneNum <31 OR PGZoneNumber> 58 THEN Tnl := Tnl + 10;
               ELSIF PGZoneNumber > 51 AND PGZoneNumber < 59 --typo in source
                      THEN Tnl := Tnl - 10;
               END IF;
       -- International
       WHEN 2=>IF PGZoneNumber  > 46 AND PGZoneNumber < 52 THEN Tnl := Tnl + 10;
               END IF;
       -- Clark 1880
       WHEN 3=>Tnl := Tnl + 10;
       -- Everest
       WHEN 4=>IF PGZoneNumber < 46 THEN Tnl := Tnl + 10; END IF;
       -- Bessel
       WHEN 5=>IF PGZoneNumber <52 AND PGZoneLetter  <'R' THEN Tnl := Tnl + 10;
               END IF;
       -- Australian
       WHEN 6=>IF ODD( PGZoneNumber ) THEN Tnl := Tnl + 15;
             END IF;
      WHEN OTHERS => NULL;    -- Not in source
   END CASE;
      --
   -- Calculate Final Northing Letter Index
   If Tnl > 14 Then Tnl := Tnl + 2;
      ELSIF Tnl > 9 THEN Tnl := Tnl + 1;
   END IF;
   Tn1 := Tn1 + CHARACTER'POS('A') - 1;
   UTMNorthingLetter := CHARACTER'VAL( Tnl );
   -- format for output
   ConvertToASCII( PGNorthingCoord MOD 100000, UTMNorthingNumber );
   ConvertToASCII( PGEastingCoord MOD 100000, UTMEastingNumber );
   UTMZoneNum(1) := PGZoneNumber / 10;
   UTMZoneNum(2) := PGZoneNumber MOD 10;
   UTMZoneLetter := PGZoneeLetter;
END ConvertToUTM;
```

```
{                                                                }
PROGRAM DRVGZTB ( INPUT,OUTPUT );
{                                                                }
{ PROGRAM DRVGZTB provides a test driver capability for testing }
{ GridZoneGeneration procedures for TRAILBLAZER.                }
{                                                                }
TYPE
   DegreesReal                      = Real;
   ZoneRange                        = 1..60;
   Letters                          = 'A'..'Z';
VAR
   Longitude                        : DegreesReal;
   Latitude                         : DegreesReal;
   GridZoneNumber                   : ZoneRange;
   GridZoneLetter                   : Letters;
{                                                                }
PROCEDURE TBGridZoneGeneration
          ({OUT}      Longitude             : DegreesReal;
           {OUT}      Latitude              : DegreesReal;
           {IN } VAR GridZoneNumber         : ZoneRange;
           {IN } VAR GridZoneLetter         : Letters);EXTERN;
{                                                                }
{ PROCEDURE TBGridZoneGeneration models the TRAILBLAZER conversion}
{ of geographic coordinates to Universal Transverse Mercator    }
{ (UTM) coordinates grid zone designator number,letter.         }
{                                                                }
   BEGIN
{                                                                }
      WRITELN (' ENTER Longitude');
      READLN ( Longitude );
      WRITELN (' ENTER Latitude');
      READLN (Latitude );
      TBGridZoneGeneration ( {OUT} Longitude,Latitude,
                             {IN } gridzonenumber,gridzoneletter);
      WRITELN;
      WRITELN (' GridZoneNumber is ',GridZoneNumber );
      WRITELN;
      WRITELN (' GridZoneLetter is ',GridZoneLetter );
{                                                                }
   END. { of PROGRAM DRVGZTB }
```

```
{                                                                   }
MODULE TBGZDG ( INPUT,OUTPUT );
{                                                                   }
TYPE
   DegreesReal                        = REAL;
   ZoneRange                          = 1..60;
   Letters                            = 'A'..'Z';
{                                                                   }
PROCEDURE TBGridZoneGeneration
          ({IN }      Longitude            : DegreesReal;
           {IN }      Latitude             : DegreesReal;
           {OUT} VAR GridZoneNumber        : ZoneRange;
           {OUT} VAR GridZoneLetter        : Letters);
{                                                                   }
{ PROCEDURE TBGridZoneGeneration models the TRAILBLAZER conversion}
{ of geographic coordinates to Universal Transverse Mercator       }
{ (UTM) coordinates - grid zone designator number, letter.         }
{                                                                   }
{ Documentation used was the GP2UM subprogram dtd. 20 feb 81       }
{ from the listings provided for the TRAILBLAZER system.           }
{                                                                   }
{ PROCEDURE TBGridZoneGeneration is referenced by:                 }
{            PROGRAM DRVGZTB                                        }
{                                                                   }
{ PROCEDURE TBGridZoneGeneration makes no references.              }
{                                                                   }
{ This procedure DOES NOT perform any data validity checks         }
{ that are not explicitly specified in the algorithm               }
{ description. This is to allow the algorithm features to be       }
{ represented more clearly.                                         }
{                                                                   }
    TYPE
       Letters                        = 'A'..'Z';
       IndexRange                     = 1..24;
{                                                                   }
    VAR
       GridZoneLtrList                : ARRAY[1..24] OF LETTERS;
       GridZoneIndex                  : IndexRange;
{                                                                   }
    BEGIN
{  Initialize allowable characters array                            }
{                                                                   }
        GridZoneLtrList [1]        :='A';
        GridZoneLtrList [2]        :='B';
        GridZoneLtrList [3]        :='C';
        GridZoneLtrList [4]        :='D';
        GridZoneLtrList [5]        :='E';
        GridZoneLtrList [6]        :='F';
        GridZoneLtrList [7]        :='G';
        GridZoneLtrList [8]        :='H';
        GridZoneLtrList [9]        :='J';
        GridZoneLtrList [10]       :='K';
        GridZoneLtrList [11]       :='L';
        GridZoneLtrList [12]       :='M';
        GridZoneLtrList [13]       :='N';
        GridZoneLtrList [14]       :='P';
        GridZoneLtrList [15]       :='Q';
        GridZoneLtrList [16]       :='R';
        GridZoneLtrList [17]       :='S';
        GridZoneLtrList [18]       :='T';
        GridZoneLtrList [19]       :='U';
        GridZoneLtrList [20]       :='V';
        GridZoneLtrList [21]       :='W';
```

```
      GridZoneLtrList [22]      :-'X';
      GridZoneLtrList [23]      :='Y';
      GridZoneLtrList [24]      :-'Z';
{                                                                 }
      GridZoneNumber  := TRUNC (31.0+(Longitude/6.0));
      GridZoneIndex   :- TRUNC (13.0+(Latitude/8.0));
      GridZoneLetter  :- GridZoneLtrList[GridZoneIndex];
{                                                                 }
END; { of PROCEDURE TBGridZoneGeneration                          }
{                                                                 }
END. { of MODULE GRGZTB                                           }
```

```
{                                                               }
PROGRAM DRVGZMG ( INPUT,OUTPUT );
{                                                               }
{ PROGRAM DRVGZMG provides a test driver capability for testing }
{ GridZoneGeneration procedures for MAGIIC.                     }
{                                                               }
TYPE
   Radians                      = REAL;
   ZoneRange                    - 1..60;
   Letters                      = 'A'..'Z';
VAR
   Longitude                    : Radians;
   Latitude                     : Radians;
   GridZoneNumber               : ZoneRange;
   GridZoneLetter               : Letters;
{                                                               }
PROCEDURE MGGridZoneGeneration
          ({OUT}     Longitude          : Radians;
           {OUT}     Latitude           : Radians;
           {IN } VAR GridZoneNumber     : ZoneRange;
           {IN } VAR GridZoneLetter     : Letters);EXTERN;
{                                                               }
{ PROCEDURE MGGridZoneGeneration models the MAGIIC conversion   }
{ of geographic coordinates to Universal Transverse Mercator    }
{ (UTM) coordinates grid zone designator number & letter.       }
{                                                               }
   BEGIN
{                                                               }
      WRITELN (' ENTER Longitude');
      READLN ( Longitude );
      WRITELN (' ENTER Latitude');
      READLN (Latitude );
      MGGridZoneGeneration ( {OUT} Longitude,Latitude,
                             {IN } GridZoneNumber,GridZoneLetter);
      WRITELN;
      WRITELN (' GridZoneNumber is ',GridZoneNumber );
      WRITELN;
      WRITELN (' GridZoneLetter is ',GridZoneLetter );
{                                                               }
   END. { of PROGRAM DRVGZMG }
```

```
{                                                                         }
MODULE MGGZDG ( INPUT,OUTPUT );
{                                                                         }
TYPE
   Radians                        = REAL;
   ZoneRange                      = 1..60;
   Letters                        = 'A'..'Z';
{                                                                         }
PROCEDURE MGGridZoneGeneration
          ({IN }     Longitude          : Radians;
           {IN }     Latitude           : Radians;
           {OUT} VAR GridZoneNumber     : ZoneRange;
           {OUT} VAR GridZoneLetter     : Letters);
{                                                                         }
{ PROCEDURE MGGridZoneGeneration models the MAGIIC conversion            }
{ of geographic coordinates to Universal Transverse Mercator             }
{ (UTM) coordinates - grid zone designator number & letter.              }
{                                                                         }
{ Documentation used was source code listings from the MAGIIC            }
{ document CG108100A dtd. 23 Oct 1978,par.3.2.90, pg.167.                }
{                                                                         }
{ PROCEDURE MGGridZoneGeneration is referenced by:                       }
{           PROGRAM DRVGZMG                                               }
{                                                                         }
{ PROCEDURE MGGridZoneGeneration makes no references.                    }
{                                                                         }
{ This procedure DOES NOT perform any data validity checks               }
{ that are not explicitly specified in the algorithm                     }
{ description. This is to allow the algorithm features to be            }
{ represented more clearly.                                               }
{                                                                         }
   CONST
      Pi                          = 3.1415926;
{                                                                         }
   TYPE
      IndexRange                  = 1..26;
{                                                                         }
   VAR
      GridZoneIndex               : IndexRange;
{                                                                         }
   BEGIN
{                                                                         }
{ Calculate the grid zone number                                          }
{                                                                         }
{ STATED LONGITUDE RANGE IS -180<-LONGITUDE<=180 IN DEGREES              }
{                                                                         }
      GridZoneNumber  := TRUNC(((180.0/Pi)*Longitude+180.0)/6.0)+1;
{                                                                         }
{ NO compensation for wrap around of grid zone numbers                   }
{                                                                         }
{ Determine the grid zone letter                                          }
{                                                                         }
{ Since no details are provided in the referenced documentation          }
{ it is ASSUMED that we know how to assign A or B for latitudes          }
{ equal to or over 84 degrees North and Y or Z for latitudes             }
{ equal to or over 80 degrees South.                                      }
{                                                                         }
{ TRUNCATION to integer is ASSUMED since it is necessary at this}
{ point in order to use the GridZoneIndex as a pointer.                  }
{                                                                         }
{ STATED LATITUDE RANGE IS 80<=LATITUDE<-84 IN DEGREES                   }
{                                                                }
      GridZoneIndex := TRUNC (((180.0/Pi)*Latitude+80.0)/8.0);


{                                                                }
{ Compute midrange grid zone letters                             }
{                                                                }
      IF GridZoneIndex <= 5
         THEN GridZoneLetter := CHR(GridZoneIndex + ORD('C'));
{ Here we handle the 'i' which is not used                       }
{                                                                }
      IF ( GridZoneIndex >= 6 ) AND
         ( GridZoneIndex <= 10 )
         THEN GridZoneLetter := CHR(GridZoneIndex + ORD('C')+1);
{                                                                }
{ Here we handle the 'O' which is not used                       }
{                                                                }
      IF ( GridZoneIndex >= 11 ) AND
         ( GridZoneIndex <= 19 )
         THEN GridZoneLetter := CHR(GridZoneIndex + ORD('C')+2);
{                                                                }
{ The rest of the GridZoneIndex are biased off by ORD ('C')      }
{                                                                }
      IF GridZoneIndex > 19
         THEN GridZoneLetter := CHR(GridZoneIndex);
{                                                                }
{ Assign Y or Z to the North Polar Zone according as Western     }
{ or Eastern Hemisphere, respectively.                           }
{                                                                }
      IF Latitude*(180.0/Pi) >= 84.0
         THEN IF Longitude*(180.0/Pi) < 0.0
                 THEN GridZoneLetter := 'Y'
                 ELSE GridZoneLetter := 'Z';
{                                                                }
{ Assign A or B to the South Polar Zone according as Western     }
{ or Eastern Hemisphere, respectively.                           }
{                                                                }
      IF Latitude*(180.0/Pi) <= -80.0
         THEN IF Longitude*(180.0/Pi) < 0.0
                 THEN GridZoneLetter := 'A'
                 ELSE GridZoneLetter := 'B';
{                                                                }
{ NO correction for the four irregular zones -                   }
{    32X,34X, and 36X do not exist                               }
{    31V is truncated                                            }
{                                                                }
END; { of PROCEDURE MGGriDZoneGeneration                         }
{                                                                }
END. { of MODULE MGGZDG                                          }
```

```
{                                                               }
PROGRAM DRVGZBT ( INPUT,OUTPUT );
{                                                               }
{ PROGRAM DRVGZBT provides a test driver capability for testing }
{ GridZoneGeneration procedures for BETA.                       }
{                                                               }
TYPE
   Radians                       = REAL;
   ZoneRange                     = 1..60;
   Letters                       = 'A'..'Z';
VAR
   Longitude                     : Radians;
   Latitude                      : Radians;
   GridZoneNumber                : ZoneRange;
   GridZoneLetter                : Letters;
   CenterMeridian                : Radians;
{                                                               }
PROCEDURE BTGridZoneGeneration
          ({OUT}      Longitude          : Radians;
           {OUT}      Latitude           : Radians;
           {IN } VAR GridZoneNumber      : ZoneRange;
           {IN } VAR GridZoneLetter      : Letters;
           {IN } VAR CenterMeridian      : Radians);EXTERN;
{                                                               }
{ PROCEDURE BTGridZoneGeneration models the BETA conversion     }
{ of geographic coordinates to Universal Transverse Mercator    }
{ (UTM) coordinates grid zone designatur number,letter and the  }
{ central meridian of the rectangle.                            }
{                                                               }
   BEGIN
{                                                               }
      WRITELN (' ENTER Longitude');
      READLN ( Longitude );
      WRITELN (' ENTER Latitude');
      READLN (Latitude );
      BTGridZoneGeneration ( {OUT} Longitude,Latitude,
                             {IN } GridZoneNumber,GridZoneLetter,
                                   CenterMeridian);
      WRITELN;
      WRITELN (' GridZoneNumber is ',GridZoneNumber );
      WRITELN;
      WRITELN (' GridZoneLetter is ',GridZoneLetter );
      WRITELN;
      WRITELN (' CenterMeridian is ',CenterMeridian )
{                                                               }
   END. { of PROGRAM DRVGZBT }
```

```
{                                                                  }
MODULE BTGZDG ( INPUT,OUTPUT );
{                                                                  }
TYPE
   Radians                           = REAL;
   ZoneRange                         = 1..60;
   Letters                           = 'A'..'Z';
{                                                                  }
PROCEDURE BTGridZoneGeneration
          ({IN }     Longitude            : Radians;
           {IN }     Latitude             : Radians;
           {OUT} VAR GridZoneNumber       : ZoneRange;
           {OUT} VAR GridZoneLetter       : Letters;
           {OUT} VAR CenterMeridian       : Radians);
{                                                                  }
{ PROCEDURE BTGridZoneGeneration models the BETA conversion        }
{ of geographic coordinates to Universal Transverse Mercator       }
{ (UTM) coordinates - grid zone designator number, letter, and     }
{ the central meridian.                                            }
{                                                                  }
{ Documentation used was source code listings from the BETA        }
{ document SS22-43 dtd. 16 Oct 1981, apx.4,pg.2-474 for the        }
{ ADSCNU subprogram and pg.2-450 for the ADSCOM subprogram         }
{                                                                  }
{ PROCEDURE BTGridZoneGeneration is referenced by:                 }
{            PROGRAM DRVGZBT                                       }
{                                                                  }
{ PROCEDURE BTGridZoneGeneration makes no references.              }
{                                                                  }
{ This procedure DOES NOT perform any data validity checks         }
{ that are not explicitly specified in the algorithm               }
{ description. This is to allow the algorithm features to be       }
{ represented more clearly.                                        }
{                                                                  }
{ Since the included 'ZDBPRO.COM' is not available to us at        }
{ this time, we assume implicit typing in the source FORTRAN       }
{ code.                                                            }
{                                                                  }
   CONST
      Pi                             = 3.1415926;
{                                                                  }
   TYPE
      Letters                        = 'A'..'Z';
      IndexRange                     = 1..24;
{                                                                  }
   VAR
      GridZoneLtrList                : ARRAY[1..24] OF LETTERS;
      GridZoneIndex                  : IndexRange;
{                                                                  }
   BEGIN
{  Initialize allowable characters array                           }
{                                                                  }
      GridZoneLtrList [1]      :='A';
      GridZoneLtrList [2]      :='B';
      GridZoneLtrList [3]      :='C';
      GridZoneLtrList [4]      :='D';
      GridZoneLtrList [5]      :='E';
      GridZoneLtrList [6]      :='F';
      GridZoneLtrList [7]      :='G';
      GridZoneLtrList [8]      :='H';
      GridZoneLtrList [9]      :='J';
      GridZoneLtrList [10]     :='K';
      GridZoneLtrList [11]     :='L';
```

```
        GridZoneLtrList [12]       :='M';
        GridZoneLtrList [13]       :='N';
        GridZoneLtrList [14]       :='P';
        GridZoneLtrList [15]       :='Q';
        GridZoneLtrList [16]       :='R';
        GridZoneLtrList [17]       :='S';
        GridZoneLtrList [18]       :='T';
        GridZoneLtrList [19]       :='U';
        GridZoneLtrList [20]       :='V';
        GridZoneLtrList [21]       :='W';
        GridZoneLtrList [22]       :='X';
        GridZoneLtrList [23]       :='Y';
        GridZoneLtrList [24]       :='Z';
{                                                                      }
{ Calculate the grid zone number                                       }
{                                                                      }
        GridZoneNumber  := (TRUNC(((18000.0/Pi)*Longitude)
                                    +18600.0)) DIV 600;
{                                                                      }
{ Compensate for wrap around of grid zone numbers                      }
{                                                                      }
        IF GridZoneNumber > 60
           THEN GridZoneNumber := GridZoneNumber-60;
        IF GridZoneNumber < 1
           THEN GridZoneNumber := GridZoneNumber+60;
{                                                                      }
{ Determine the grid zone letter                                       }
{                                                                      }
        GridZoneIndex := TRUNC((Latitude*(180.0/Pi)+104.0)/8.0);
{                                                               }
{ Test for and lock out North Polar Zones                              }
{                                                                      }
        IF GridZoneIndex > 22

           THEN GridZoneIndex := 22;
{                                                                      }
{ NOTE that no such test is needed for the South Polar Zone            }
{ because the algoritm limit was given as <= 80 South                  }
{                                                                      }
        GridZoneLetter := GridZoneLtrList[GridZoneIndex];
{                                                                      }
{ Correct for the four irregular zones -                               }
{     32X,34X, and 36X do not exist                                    }
{     31V is truncated                                                 }
{                                                                      }
{ Truncate grid zone 31V                                               }
{                                                                      }
        IF (GridZoneIndex = 20) AND
           ((GridZoneNumber =31) AND
           (Longitude >= 3.0*(Pi/180.0)))
           THEN GridZoneNumber := 32;
{                                                                      }
{ Correct for grid zones 32X,34X, and 36X                              }
{                                                                      }
        IF (GridZoneIndex =22) AND
           (GridZoneNumber =32)
           THEN IF Longitude >= 9.0*(Pi/180.0)
                   THEN GridZoneNumber := 33
                   ELSE GridZoneNumber := 31;
{                                                                      }

        IF (GridZoneIndex = 22) AND
           (GridZoneNumber = 34)
```

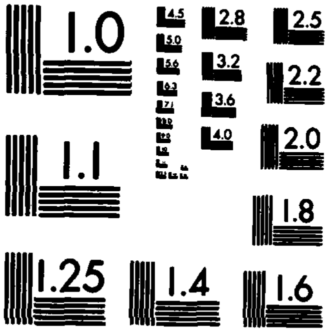
```
         THEN IF Longitude >= 21.0*(Pi/180.0)

                 THEN GridZoneNumber := 35
                 ELSE GridZoneNumber := 33;
{                                                                }
      IF (GridZoneIndex =22) AND
         (GridZoneNumber = 36)
         THEN IF Longitude >= 33.0*(Pi/180.0)
                 THEN GridZoneNumber := 37
                  ELSE GridZoneNumber := 35;
{                                                                }
      CenterMeridian  := (6*GridZoneNumber-183)*(Pi/180.0)
{                                                                }
end; { of PROCEDURE BTGriDZoneGeneration
{                                                                }
END. { of MODULE BTGZDG                                          }
```

```
{                                                                      }
PROGRAM DRVGZGR ( INPUT,OUTPUT );
{                                                                      }
{ PROGRAM DRVGZGR provides a test driver capability for testing }
{ GridZoneGeneration procedures for GUARDRAIL.                         }
{                                                                      }
TYPE
   DegreesReal                        = Real;
   DegreesInteger                     = INTEGER;
   ZoneRange                          = 1..60;
   Letters                            = 'A'..'Z';
VAR
   Longitude                          : DegreesReal;
   Latitude                           : DegreesReal;
   GridZoneNumber                     : ZoneRange;
   GridZoneLetter                     : Letters;
   centermeridian                     : degreesinteger;
{                                                                      }
PROCEDURE GRGridZoneGeneration
           ({OUT}      Longitude            : DegreesReal;
            {OUT}      Latitude             : DegreesReal;
            {IN } VAR GridZoneNumber       : ZoneRange;
            {IN } VAR GridZoneLetter       : Letters;
            {IN } VAR CenterMeridian       : DegreesInteger);EXTERN;
{                                                                      }
{ PROCEDURE GRGridZoneGeneration models the GUARDRAIL conversion}
{ of geographic coordinates to Universal Transverse Mercator    }
{ (UTM) coordinates grid zone designator number,letter and the  }
{ central meridian of the rectangle.                            }
{                                                                      }
   BEGIN
{                                                                      }
      WRITELN (' ENTER Longitude');
      READLN ( Longitude );
      WRITELN (' ENTER Latitude');
      READLN (Latitude );
      GRGridZoneGeneration ( {OUT} Longitude,Latitude,
                             {IN } gridzonenumber,gridzoneletter,
                                   centermeridian);
      WRITELN;
      WRITELN (' GridZoneNumber is ',GridZoneNumber );
      WRITELN;
      WRITELN (' GridZoneLetter is ',GridZoneLetter );
      WRITELN;
      WRITELN (' CenterMeridian is ',CenterMeridian )
{                                                                      }
   END. { of PROGRAM DRVGZGR }
```

```
{                                                                }
MODULE GRGZDG ( INPUT,OUTPUT );
{                                                                }
TYPE
   DegreesReal                     = REAL;
   DegreesInteger                  = INTEGER;
   ZoneRange                       = 1..60;
   Letters                         = 'A'..'Z';
{                                                                }
PROCEDURE GRGridZoneGeneration
          ({IN }      Longitude          : DegreesReal;
           {IN }      Latitude           : DegreesReal;
           {OUT} VAR GridZoneNumber      : ZoneRange;
           {OUT} VAR GridZoneLetter      : Letters;
           {OUT} VAR CenterMeridian      : DegreesInteger);
{                                                                }
{ PROCEDURE GRGridZoneGeneration models the GUARDRAIL conversion}
{ of geographic coordinates to Universal Transverse Mercator    }
{ (UTM) coordinates - grid zone designator number, letter and   }
{ the central meridian.                                         }
{                                                                }
{ PROCEDURE GRGridZoneGeneration is referenced by:              }
{            PROGRAM DRVGZGR                                    }
{                                                                }
{ PROCEDURE GRGridZoneGeneration makes no references.           }
{                                                                }
{ This procedure DOES NOT perform any data validity checks      }
{ that are not explicitly specified in the algorithm            }
{ description. This is to allow the algorithm features to be    }
{ represented more clearly.                                     }
{                                                                }
   TYPE
      Letters                      = 'A'..'Z';
      IndexRange                   = 1..24;
{                                                                }
   VAR
      GridZoneLtrList              : ARRAY[1..24] OF LETTERS;
      GridZoneIndex                : IndexRange;
{                                                                }
   BEGIN
{  Initialize allowable characters array                         }
{                                                                }
      GridZoneLtrList [1]          :='A';
      GridZoneLtrList [2]          :='B';
      GridZoneLtrList [3]          :='C';
      GridZoneLtrList [4]          :='D';
      GridZoneLtrList [5]          :='E';
      GridZoneLtrList [6]          :='F';
      GridZoneLtrList [7]          :='G';
      GridZoneLtrList [8]          :='H';
      GridZoneLtrList [9]          :='J';
      GridZoneLtrList [10]         :='K';
      GridZoneLtrList [11]         :='L';
      GridZoneLtrList [12]         :='M';
      GridZoneLtrList [13]         :='N';
      GridZoneLtrList [14]         :='P';
      GridZoneLtrList [15]         :='Q';
      GridZoneLtrList [16]         :='R';
      GridZoneLtrList [17]         :='S';
      GridZoneLtrList [18]         :='T';
      GridZoneLtrList [19]         :='U';
      GridZoneLtrList [20]         :='V';
      GridZoneLtrList [21]         :='W';
```

```
        GridZoneLtrList [22]        :='X';
        GridZoneLtrList [23]        :='Y';
        GridZoneLtrList [24]        :='Z';
{                                                                  }
        GridZoneNumber  := TRUNC (31.0+(Longitude/6.0));
        GridZoneIndex   := TRUNC (13.0+(Latitude/8.0));
        GridZoneLetter  := GridZoneLtrList[GridZoneIndex];
        CenterMeridian  := 6*GridZoneNumber-183
{                                                                  }
END; { of PROCEDURE GRGriDZoneGeneration                                  }
{                                                                  }
END. { of MODULE GRGZDG                                            }
```

Cont



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

# SUPPLEMENTARY

# INFORMATION

Corrected DD 1473

AD-A129 182

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

## REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
D-181

2. GOVT ACCESSION NO.
AD A129 182

3. RECIPIENT'S CATALOG NUMBER

4. TITLE *(and Subtitle)*
Analysis of Geographic Transformation Algorithms

5. TYPE OF REPORT & PERIOD COVERED
FINAL

6. PERFORMING ORG. REPORT NUMBER
D-181

7. AUTHOR(s)
J. Gillis, J. Radbill, A. Griesel, N. Palmer, P. Babby

8. CONTRACT OR GRANT NUMBER(s)
NAS7-918

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Jet Propulsion Laboratory ATTN:171-209
California Institute of Technology
4800 Oak Grove, Pasadena, CA 91109

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
RE 182
AMEND # 187

11. CONTROLLING OFFICE NAME AND ADDRESS
Commander, USAICS
ATTN: ATSI-CD-SF
Ft. Huachuca, AZ 85613-7000

12. REPORT DATE
July 9, 1982

13. NUMBER OF PAGES
35

14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*
(Same as Item 11)

15. SECURITY CLASS. *(of this report)*
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
NONE

16. DISTRIBUTION STATEMENT *(of this Report)*
Approved for Public Dissemination

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES
Prepared by Jet Propulsion Laboratory for the US Army Intelligence Center and School's Combat Developer's Support Facility

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*
GEOGRAPHIC TRANSFORMATION, COORDINATE TRANSFORMATION, ALGORITHMS, MAGIIC, BETA, GUARDRAIL, TRAILBLAZER, ALGORITHM ANALYSIS, SOFTWARE ANALYSIS, IEW SYSTEMS

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This report describes the findings of JPL regarding geographic transformation algorithms used in MAGIIC, GUARDRAIL, TRAILBLAZER and BETA systems. A set of parameters is developed to characterize and catalogue intelligence system algorithms in the four systems. Individual algorithms are also analyzed to determine if they are performing their functions properly.